UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIENCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcos Alexandre Miguel

GIVEME EFFORT: UM *FRAMEWORK* PARA APOIAR ESTIMATIVA DE ESFORÇO EM ATIVIDADES DE MANUTENÇÃO E COMPREENSÃO DE SOFTWARE

Juiz de Fora
2016

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIENCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcos Alexandre Miguel

GIVEME EFFORT: UM *FRAMEWORK* PARA APOIAR ESTIMATIVA DE ESFORÇO EM ATIVIDADES DE MANUTENÇÃO E COMPREENSÃO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. D.Sc. José Maria Nazar David

Juiz de Fora
2016

Marcos Alexandre Miguel

GIVEME EFFORT: UM *FRAMEWORK* PARA APOIAR ESTIMATIVA DE ESFORÇO EM ATIVIDADES DE MANUTENÇÃO E COMPREENSÃO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 01 de Setembro de 2016.

BANCA EXAMINADORA

_____

Prof. D.Sc. José Maria Nazar David - Orientador
Universidade Federal de Juiz de Fora

_____

Prof. D.Sc. Marco Antônio Pereira Araújo – Coorientador
Universidade Federal de Juiz de Fora

_____

Prof. D.Sc. Regina Maria Maciel Braga
Universidade Federal de Juiz de Fora

_____

Prof. D.Sc. Glauco de Figueiredo Carneiro
Universidade Salvador (UNIFACS)

AGRADECIMENTOS

Primeiramente ao meu bom Deus Jeová, por me dotar da força irresistível de buscar meus sonhos e ideias, pelo dom da vida e por sempre estar ao meu lado em todos os momentos da minha vida e ter me guiado até aqui.

À minha esposa Loanda por todo o carinho, compreensão e ajuda durante essa caminhada. Obrigado por você ser meu complemento, minha força e ajuda nos momentos mais difíceis que passei e que foram decisivos para que eu pudesse ter serenidade, hombridade e fazer o que era certo. Sem você nada disso poderia ser possível.

Aos meus pais, Luiz e Sueli Miguel, por toda educação, senso moral e exemplo e por sempre me incentivarem a buscar meus ideais, por compreenderem minha ausência e por mesmo assim estarem sempre ao meu lado. Meras palavras são insuficientes para agradecer a vida e a educação que me deram. Minha fibra moral e o que sou refletem um pouco da forma como me educaram.

Aos amigos Humberto, Welington, Tassio, Jacimar (*Forever Alone*) e muitos outros, pela incansável torcida por meu sucesso.

Ao professor e orientador José Maria N. David, pelo conhecimento passado, compreensão, dedicação, atenção, paciência, pela sua iniciativa e comprometimento ao meu trabalho. Sua nobre educação e tato junto com suas observações foram fundamentais para que eu conseguisse concluir esse trabalho. Entrei no programa de mestrado como seu orientando e vou sair dele como seu amigo.

Ao professor, orientador e grande amigo, Marco Antônio Pereira Araújo, pela compreensão, paciência, dedicação, por estar ao meu lado sempre que precisei e por inúmeras vezes em minha vida, abrir novas oportunidades para meu crescimento pessoal e profissional. Nada disso seria possível, se você não fosse essa pessoa generosa e amiga. Muito do que sou agora, em sentido profissional, eu devo a você, meu nobre amigo, e espero que consiga transmitir o seu legado a muitos outros que eu possa ajudar futuramente.

A todos os professores, por tudo que me ajudaram meu simples, mas eterno obrigado.

"Dentro da noite que me rodeia, negra como um poço de lado a lado, Eu agradeço a Deus, por minha alma indomável. Nas garras cruéis da circunstância eu não tremo ou me desespero. Sob os duros golpes da sorte, minha cabeça sangra, mas não se curva. Além deste lugar de raiva e choro paira somente o horror da sombra. E ainda assim, a ameaça do tempo vai me encontrar, e deve me achar destemido.
Não me importa se o portão é estreito,
Não me importa o tamanho do castigo,
Eu sou o dono do meu destino,
Eu sou o capitão de minha alma."
Wiliam Ernet Henley

"Se eu falar em línguas de homens e de anjos e entender todos os segredos e todo o conhecimento, mas não tiver amor, nada sou."
1 Coríntios 13:1-2

RESUMO

Muitas organizações encontram problemas na tentativa de estimar esforço em atividades de manutenção de software. Quando a estimativa de esforço não está bem definida ou é imprecisa, os resultados obtidos podem refletir diretamente na entrega do software, causando insatisfação do cliente ou diminuição da qualidade do produto. O sucesso ou fracasso de projetos depende da precisão do esforço e do cronograma das atividades envolvidas. O surgimento de métodos ágeis no campo de desenvolvimento de software tem apresentado muitas oportunidades e desafios para pesquisadores e profissionais da área. Um dos principais desafios é a estimativa de esforço para as atividades de manutenção no desenvolvimento ágil de software. Nesse contexto, este trabalho apresenta um *framework*, nomeado *GiveMe Effort*, o qual objetiva apoiar as atividades de estimativa de esforço na manutenção de software usando dados históricos e informações de compreensão de software.

Palavras-chave: Estimativa de Esforço, Manutenção de Software, Compreensão de Software, Visualização de Software.

ABSTRACT

Many organizations encounter problems when estimating effort for software maintenance activities. When estimating effort is not well defined or are inaccurate, the results may reflect directly into the software delivery, causing customer dissatisfaction or decreased product quality. The success or failure of projects depends on the accuracy of the effort and the schedule of involved activities. The rise of agile methods in software development has presented many opportunities and challenges for researchers and professionals. In this context, a key challenge is the effort estimate for maintenance activities in the agile software development context. This work presents a *framework*, called GiveMe Effort, to support the effort estimation activities in software maintenance. It is based on historical data and software comprehension information.

Keywords: Effort Estimation, Software Maintenance, Software Comprehension, Software Visualization.

LISTA DE FIGURAS

Figura 1: Estrutura do trabalho 18
Figura 2: Visão geral do diagrama de componentes do *GiveMe Infra* e a integração da solução atual a infraestrutura (TAVARES, 2015) 34
Figura 3: Visão geral da arquitetura do *GiveMe Effort* 35
Figura 3: Exemplo de aplicação da distância de Levenshtein 39
Figura 4: Resultado da aplicação da distância de Levenshtein 40
Figura 6: Norma Euclidiana de um vetor 41
Figura 7: Interseção entre conjuntos nas técnicas apresentadas pelo *framework* 42
Figura 8: Regressão Linear – Representação Gráfica (MONTGOMERY, 2012) 43
Figura 9: Visualização *Pie Chart* View 46
Figura 10: Área representativa do Gráfico 46
Figura 11: Visualização *Bubble View* 47
Figura 12: Visualização *Interactive Force View* 48
Figura 13: Visualização *Veen Diagram View* 49
Figura 14: Visualização *Box Plot View* 50
Figura 15: Visualização *Reputation Developers View* 51
Figura 16: Visualização *Linear Regression View* 52
Figura 17: Visualização *Code Analyser View* 53
Figura 18: Componente *Planning* 54
Figura 19: Exemplo de Requisição de Mudança 55
Figura 20: *GiveMe Effort* - Opções de configuração e calibração do *framework* 56
Figura 21: *GiveMe Effort* - Passo 1 – Análise Textual 57
Figura 22: Análise dos dados da requisição de mudança 57
Figura 23: Guia *Filter Analysis* 58
Figura 24: Visualização *Pie Chart* gerada pelo exemplo de funcionamento da proposta 59
Figura 25: Visualização *Linear Regression* gerada pelo exemplo de funcionamento da proposta 59
Figura 26: Visualização *Bubble View* gerada pelo exemplo de funcionamento da proposta 60
Figura 27: Visualização *Reputation Developers View* gerada pelo exemplo de funcionamento da proposta 61
Figura 28: Visualização *Veen Diagrama View* gerada pelo exemplo de funcionamento da proposta 62
Figura 29: Visualização *Code Analyser* gerada pelo exemplo de funcionamento da proposta 63

Figura 30: Guia *Result* do *framework* gerada pelo exemplo de funcionamento da proposta 64
Figura 31: Estimativa x Tempo Gasto real da solicitação analisada 65
Figura 32: Fluxo de funcionamento geral da arquitetura do *GiveMe Effort* 66
Figura 32: MEDIÇÃO 1 - Gráfico comparativo das técnicas 70
Figura 34: MEDIÇÃO 2 - Gráfico comparativo das técnicas 72
Figura 35: MEDIÇÃO 3 - Gráfico comparativo das técnicas 74
Figura 36: MEDIÇÃO 4 - Gráfico comparativo das técnicas 76
Figura 37: Comparativo entre as medições realizadas 78
Figura 38: Comparativo entre as técnicas 78
Figura 39: Gráfico de área comparativo entre as medições realizadas 79
Figura 40: Hierarquia do modelo GQM 80
Figura 41: Idade dos participantes 83
Figura 42: Profissão dos participantes 84
Figura 43: Escolaridade dos participantes 84
Figura 44: Cursos dos participantes 85
Figura 45: Conhecimento sobre Estimativa de Esforço 85
Figura 46: Conhecimento sobre atividades de manutenção e compreensão de software 86
Figura 47: Calibração realizada pelo participante 88
Figura 48: Resultados das estimativas realizadas pelo participante 89
Figura 49: Resultados da análise qualitativa das técnicas realizada pelo participante ... 89
Figura 50: Escala utiliza para mensuração qualitativa do experimento 90
Figura 51: Resultados da análise qualitativa das Visualizações pelo participante 90
Figura 52: Calibração realizada pelo participante 90
Figura 53: Resultados da análise qualitativa das técnicas realizada pelo participante ... 91
Figura 54: Resultados da análise qualitativa das Técnicas pelo participante 91
Figura 55: Resultados da análise qualitativa das visualizações pelo participante 92
Figura 56: Calibração realizada pelo participante 92
Figura 57: Resultados da análise qualitativa das técnicas realizada pelo participante ... 93
Figura 58: Resultados da análise qualitativa das Visualizações pelo participante 94
Figura 59: Resultados da análise qualitativa das visualizações pelo participante 94
Figura 60: Calibração realizada pelo participante 95
Figura 61: Resultados das estimativas realizadas pelo participante 95
Figura 62: Resultados da análise qualitativa das técnicas realizada pelo participante ... 96
Figura 63: Resultados da análise qualitativa das Visualizações pelo participante 97
Figura 64: Calibração realizada pelo participante 97
Figura 65: Resultados das estimativas realizadas pelo participante 98
Figura 66: Resultados da análise qualitativa das técnicas realizada pelo participante ... 99
Figura 67: Resultados da análise qualitativa das visualizações pelo participante 99
Figura 68: Calibração e observações realizadas pelo participante 100
Figura 69: Resultados das estimativas realizadas pelo participante 101
Figura 70: Resultados da análise qualitativa das técnicas realizada pelo participante . 101
Figura 71: Resultados da análise qualitativa das Visualizações pelo participante 102
Figura 72: Comparativo gráfico de participantes em função das técnicas 104

Figura 73: Comparativo gráfico entre as técnicas por participante 104
Figura 74: *Boxplot* das técnicas em função do Tempo Gasto e Estimado PMG 106
Figura 75: Teste de normalidade sobre a variável Tempo. 107
Figura 76: Teste de igualdade de variância sobre a variável Tempo 107
Figura 77: Análise do teste paramétrico ANOVA 108
Figura 78: Análise gráfica do teste paramétrico ANOVA 109
Figura 79: Gráfico de análise qualitativa das técnicas. 110
Figura 80: Comparativo de análise qualitativa das visualizações. 111
Figura 81: Comparativo de análise qualitativa das visualizações por desenvolvedor. . 112

LISTA DE TABELAS

Tabela 1: Tabela Comparativa entras as Ferramentas/*Frameworks* 30
Tabela 2: Requisições de mudança e estimativa de tempo calculado pela equipe da empresa parceira 68
Tabela 3: Configuração - medição 69
Tabela 4: Medição 1 – Comparação das Técnicas 69
Tabela 5: Medição 2 – calibração 71
Tabela 6: Medição 2 – Comparação das Técnicas 71
Tabela 7: Medição 3 – calibração 73
Tabela 8: Medição 3 – comparação das técnicas 73
Tabela 9: Medição 4 – calibração da ferramenta 75
Tabela 10: Medição 4 – comparação das técnicas 75
Tabela 11: Comparação das técnicas em função das medições 77
Tabela 12: Requisições de mudança e valores encontrados ao utilizar o *framework* 87
Tabela 13: Requisições de mudança e as notas das avaliações qualitativas ao utilizar o *framework* 88
Tabela 14: Comparação das técnicas em função das medições 103
Tabela 15: Comparação das análises qualitativas das técnicas 110
Tabela 16: Comparação das análises qualitativas das visualizações 111

Sumário

1. INTRODUÇÃO 14
   1.1 MOTIVAÇÃO 15
   1.2 PROBLEMA 15
   1.3 OBJETIVO 15
   1.4 ENFOQUE DA SOLUÇÃO 16
   1.5 HIPÓTESE 16
   1.6 METODOLOGIA 16
   1.7 ESTRUTURA DA DISSERTAÇÃO 17
2. PRESSUPOSTOS TEÓRICOS 19
   2.1 MANUTENÇÃO DE SOFTWARE 20
   2.2 COLABORAÇÃO EM SOFTWARE 20
   2.3 VISUALIZAÇÃO DE SOFTWARE 22
   2.4 ESTIMATIVA DE TEMPO 23
   2.5 REQUISIÇÃO DE MUDANCAS 24
   2.6 RASTREABILIDADE 25
   2.7 TRABALHOS RELACIONADOS 25
   2.8 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO 31
3. GIVEME EFFORT 32
   3.1 ARQUITETURA 35
   3.2 NÚCLEO – SUMO METRICS MODULE 38
   3.3 NÚCLEO – LEVENSHTEIN MODULE 39
   3.4 NÚCLEO – REPUTATION MODULE 40
   3.5 NÚCLEO - INTERSECTION MODULE 41
   3.6 NÚCLEO – LINEAR REGRESSION MODULE 42
   3.7 VIEW ZONE 45
   3.8 PIE CHART VIEW 46
   3.9 BUBBLE VIEW 47
   3.10 INTERACTIVE FORCE VIEW 48
   3.11 VEEN DIAGRAM VIEW 49

3.12 BOX PLOT VIEW 50
3.13 REPUTATION DEVELOPERS VIEW 51
3.14 LINEAR REGRESSION VIEW 52
3.15 CODE ANALYSER VIEW 52
3.16 PLANNING 53
3.17 EXEMPLO DE USO DA SOLUÇÃO PROPOSTA 54
3.18 CONSIDERAÇÕES FINAIS DO CAPÍTULO 66
4. GIVEME EFFORT – FUNCIONAMENTO DO FRAMEWORK E MEDIÇÕES 67
4.1 FUNCIONAMENTO DO FRAMEWORK 67
4.2 MEDIÇÕES 69
4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO 79
5. AVALIAÇÃO DA SOLUÇÃO 80
5.1 PLANEJAMENTO DO ESTUDO EXPERIMENTAL 80
5.2 CONDUÇÃO DO ESTUDO EXPERIMENTAL 82
5.3 TERMO DE CONSENTIMENTO DOS PARTICIPANTES 82
5.4 CARACTERIZAÇÃO DOS PARTICIPANTES 82
5.5 ESTUDO EXPERIMENTAL 87
5.6 AMEAÇAS À VALIDADE 113
5.7 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO 114
6. CONCLUSÕES 115
6.1 DIFICULDADES ENCONTRADAS 116
6.2 TRABALHOS FUTUROS 116
REFERÊNCIAS 118

1. INTRODUÇÃO

Para manter um software é necessário planejar (PRIKLADNICKI & AUDY, 2007), definir estratégias a serem adotadas ao longo do ciclo de desenvolvimento. Nesse contexto é essencial a elaboração de um processo de estimativa de esforço de software, sendo este um ponto crítico no ciclo de desenvolvimento, haja vista que o sucesso ou fracasso de projetos depende da precisão do esforço e do cronograma das estimativas. A referida estimativa é um dos principais desafios para o desenvolvimento ágil de software, pois as abordagens tradicionais têm apresentado resultados imprecisos, segundo ZIA et al. (2012).

Quando a estimativa de esforço não ocorre nos projetos de uma empresa, a visão geral do andamento e a entrega dos projetos podem ficar comprometidas. Além disso, a falta da estimativa de esforço pode apresentar uma dimensão equivocada do projeto, tal como a atribuição imprecisa de tempo para realização das atividades de manutenção de software. Informações importantes ao desenvolvimento e conclusão dos projetos podem ser perdidas devido a falta de planejamento efetivo, dificultando, ou até mesmo inviabilizando, uma visão de desenvolvimento futura (PRIKLADNICKI e AUDY, 2007). Adicionalmente, devido à evolução e manutenção constantes em um software, torna-se fundamental a construção de teorias e modelos que permitam a compreensão do passado, presente e futuro do ciclo de manutenção (PAREDES, 2014).

A estimativa de esforço envolve o apoio à equipe nas manutenções corretivas, adaptativas ou evolutivas. Estimativas de esforço, no planejamento das atividades de manutenção de software, definidas de forma *ad-hoc*, são imprecisas e podem impactar diretamente na entrega e qualidade do software (PRIKLADNICKI e AUDY, 2007).

Informações históricas, armazenadas ao longo do ciclo de vida de um software, podem auxiliar as equipes a criar mecanismos para mensurar tempo e esforço das atividades a serem desenvolvidas (PONCIN, 2011). Essas medidas, caso sejam tratadas de forma correta, podem permitir a adoção de iniciativas que objetivem melhorar a qualidade do software, bem como auxiliar equipes no dimensionamento de suas atividades (TAVARES et al., 2015). Para um acompanhamento adequado dessas atividades, uma quantidade significativa de dados deve ser coletada, processada e armazenada ao longo do tempo. Estes dados devem ser apresentados em uma interface

que promova interação com o desenvolvedor (WERNER et al., 2011). Um Ambiente Interativo baseado em Múltiplas Visões (AIMV) (CARNEIRO et al., 2012) oferece recursos e mecanismos de visualização para apoiar a análise de dados e também a descoberta e o reconhecimento de informação relevante.

Mediantes os conceitos acima relacionados, segue-se a motivação, bem como o problema levantado, o enfoque da solução encontrada, o objetivo e por fim, a hipótese que permeou todo o trabalho.

1.1 MOTIVAÇÃO

A motivação do respectivo trabalho surge da busca pela melhoria no processo de estimativa de esforço, através da sua automatização por meio do desenvolvimento de um *framework* que apresente resultados relevantes para as equipes de desenvolvimento, integrado às atividades de compreensão de software em um AIMV, e que forneça elementos de visualização em um ambiente de desenvolvimento distribuído de software. Não se encontrou, na literatura, soluções que englobem os temas de Estimativa de Esforço, Manutenção de Software, Compreensão de Software, Visualização de Software, abordados no contexto desta pesquisa.

1.2 PROBLEMA

O problema tratado neste trabalho se refere à falta de uma solução (técnica, metodologia, ferramenta, *framework* ou modelo) que apoie a estimativa de esforço no contexto de manutenção e compreensão de software, para equipes geograficamente distribuídas que utilizam metodologias de desenvolvimento ágeis.

1.3 OBJETIVO

Mediante ao problema geral apresentado na seção anterior, definiu-se como objetivo o desenvolvimento de um *framework* para apoiar a estimativa de esforço, baseado no histórico de mudanças ocorridas, interligadas ao processo de compreensão e manutenção de software. Objetiva-se ainda prover mecanismos para planejamento e gestão de atividades de manutenção (adaptativa, evolutiva e corretiva), que ofereça elementos de visualização em um ambiente de desenvolvimento distribuído de software.

1.4 ENFOQUE DA SOLUÇÃO

A solução desenvolvida foi focada no tratamento da Estimativa de Esforço em um AIMV (SILVA et al., 2012), considerando a compreensão e a manutenção de software, apoiadas pela extração de dados históricos em projetos com metodologias ágeis de desenvolvimento em um ambiente distribuído de software.

1.5 HIPÓTESE

Mediante o objeto apresentado a seguinte hipótese foi estabelecida para este trabalho após a definição do *GiveMe Effort*:

- H $_{nula}$: *GiveMe Effort* **não** é capaz de auxiliar equipes na tarefa de estimativa de esforço, auxiliando nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis em um ambiente distribuído de software;
- H $_{alternativa}$: *GiveMe Effort* **é** capaz de auxiliar equipes na tarefa de estimativa de esforço, auxiliando nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis em um ambiente distribuído de software.

1.6 METODOLOGIA

A metodologia adotada neste trabalho é especificada abaixo, em etapas, iniciando-se após a motivação o enfoque e a descoberta do problema apresentado.

- Etapa (i): Realização de um levantamento bibliográfico;
- Etapa (ii): Análise das publicações obtidas como retorno na revisão sistemática da literatura, com o objetivo de conhecer o estado da arte das pesquisas na área de Estimativa de Esforço, em termos de tecnologias utilizadas, e investigar os conceitos capazes de apoiar a resolução do problema geral deste trabalho;
- Etapa (iii): Documentação dos trabalhos relacionados, obtidos com base na revisão sistemática de literatura mencionada na Etapa (i);
- Etapa (iv): Planejamento e desenvolvimento do *framework GiveMe Effort*;
- Etapa (v): Neste ponto foi definida a hipótese geral deste trabalho, podendo ser aceita (H1) ou rejeitada (H0) com base nos resultados do estudo experimental realizado;

- Etapa (vi): Definição e avaliação do estudo experimental;
- Etapa (vii): Documentação dos resultados obtidos pelo estudo experimental;
- Etapa (viii): Validação da hipótese do trabalho;
- Etapa (ix): Elaboração da documentação do conteúdo gerado neste trabalho.

1.7 ESTRUTURA DA DISSERTAÇÃO

Este trabalho está dividido em seis capítulos ilustrados na Figura 1. O capítulo 1 apresenta uma visão geral do problema que está sendo tratado, o objetivo do trabalho, a solução desenvolvida e a metodologia utilizada. O capítulo 2 apresenta os pressupostos teóricos que fornecem a base para este trabalho. O capítulo 3 apresenta os detalhes do desenvolvimento do *framework GiveMe Effort*. O capítulo 4 apresenta o *GiveMe Effort* em ação, ou seja, uma demonstração de utilização do *framework* em um contexto real. O capítulo 5 apresenta a avaliação do *framework*, realizada por meio de estudo experimental. Por fim, o capítulo 6 apresenta as conclusões acerca do projeto.

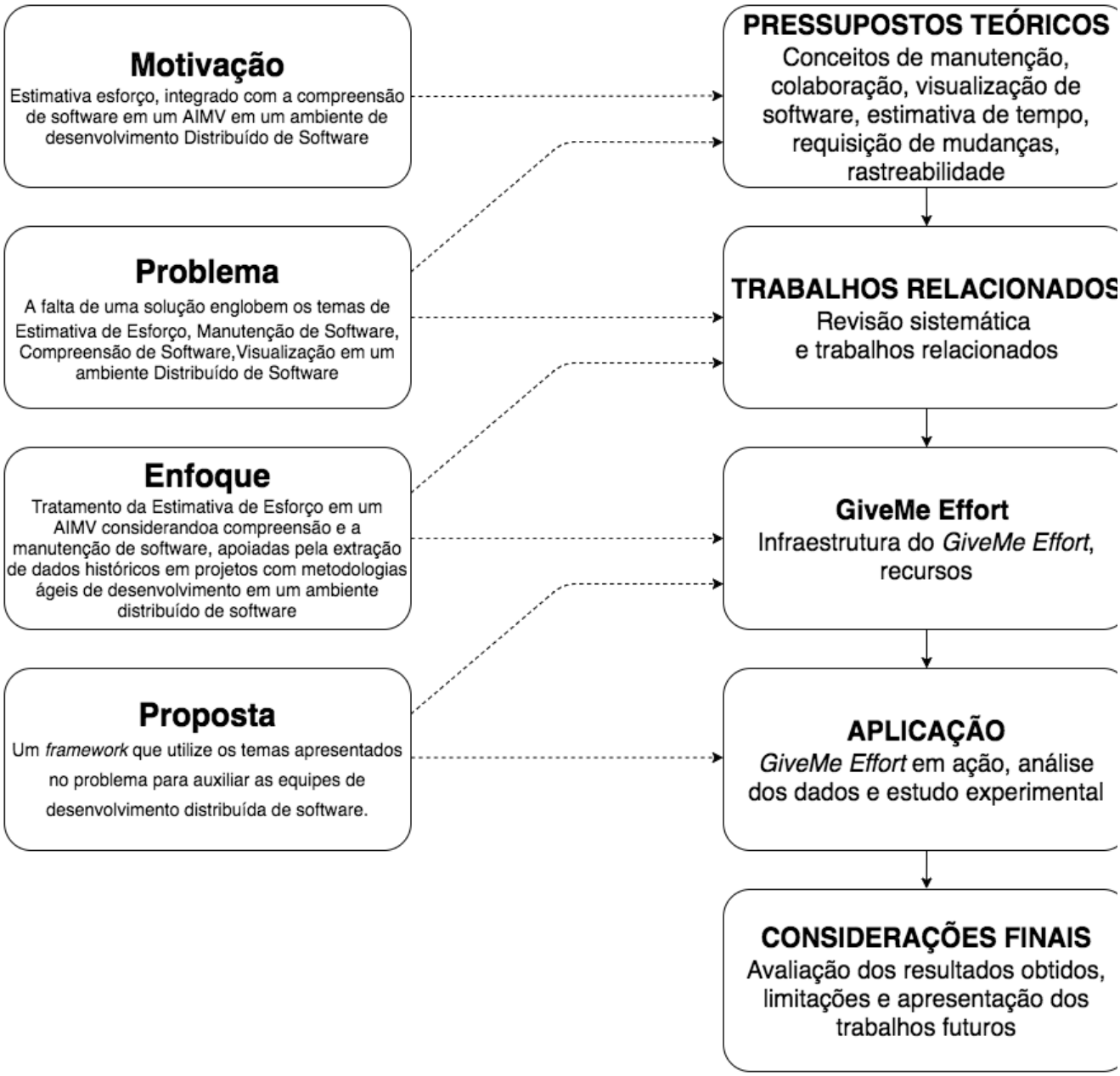


Figura 1: Estrutura do trabalho

2. PRESSUPOSTOS TEÓRICOS

Este capítulo apresenta os conceitos base deste trabalho, bem como apresenta uma visão geral das relações existentes entre eles, dado que o problema geral abordado tem como proposta de solução o desenvolvimento de uma ferramenta que combina os seguintes temas: estimativa de esforço, manutenção e compreensão de software, e visualização de software.

A pesquisa realizada para desenvolvimento do respectivo trabalho inicia-se a partir da identificação do cenário descrito, envolvendo a observação empírica do problema apresentado.

Foi realizada uma revisão sistemática da literatura (MIGUEL et al., 2015), no intuito de identificar publicações que possam indicar soluções que combinem os seguintes temas: (i) planejamento de software; (ii) rastreabilidade de software (iii) requisições de mudanças em software; (iv) estimativa de software e (v) colaboração em software. Os temas abordados visam buscar a literatura relevante em relação ao uso de estimativas na manutenção de software, com base na questão primária de pesquisa definida: quais são as técnicas/métodos de estimativa de esforço que estão sendo utilizadas no contexto de manutenção de software em métodos ágeis.

Faz-se necessário ressaltar que, através da revisão realizada, não foi possível encontrar soluções que englobem todos os temas pesquisados, o que levou à formulação do problema geral tratado neste trabalho, que se refere à falta de uma solução (técnica, metodologia, ferramenta, *framework* ou modelo) que apoie a estimativa de esforço, no contexto de manutenção e compreensão de software, para equipes geograficamente distribuídas, utilizando metodologias de desenvolvimento ágeis. Reitera-se que o termo "soluções existentes" refere-se às soluções (técnicas, metodologias, ferramentas, *frameworks* ou modelos) encontradas através da revisão sistemática de literatura realizada.

As subseções deste capítulo apresentarão os principais conceitos da proposta, e a penúltima subseção apresenta os detalhes da revisão sistemática realizada. Por fim, são dispostas as considerações finais do capítulo.

2.1 MANUTENÇÃO DE SOFTWARE

Sistemas de software passam por mudanças e adaptações ao longo do tempo. Dentre os principais motivos destacam-se as correções de erros, adaptação a novas plataformas e melhorias, visando o aumento do desempenho e atendimento a novos requisitos do software. Para a realização dessas atividades, existe a necessidade de planejamento por parte das equipes de manutenção do software (RAJLICH, 2014).

Existem diferentes significados para os tipos de manutenção encontrados na literatura. No contexto deste artigo, manutenção corretiva é o tipo de manutenção voltada para correção de erros encontrados no software. Já manutenção adaptativa refere-se a modificações que visam adequar o software a um novo ambiente, por exemplo, o suporte a uma nova plataforma. A manutenção evolutiva pode significar melhoria do projeto que visa agregar novas funcionalidades, além de melhorias para os usuários que as solicitaram. Como resultado, o projeto torna-se mais coerente ao que foi especificado pela equipe que o desenvolveu e às necessidades previamente analisadas (RAJLICH, 2014).

A próxima seção tratará do tema Colaboração de Software visto que o mesmo está permeando os conceitos do presente trabalho e foi um dos assuntos que motivaram a pesquisa.

2.2 COLABORAÇÃO EM SOFTWARE

A colaboração possibilita que grupos (equipes de manutenção, por exemplo) se organizem de forma em que haja predominância de elementos como cooperação, comunicação e coordenação (FUKS et al., 2003).

A percepção pode ser considerada um elemento de colaboração, dado que se refere às informações das interações realizadas entre membros de uma equipe. Como a manutenção de software pode ser realizada de forma colaborativa (D'AMBROS et al., 2003), então é possível obter elementos de percepção associados a ela, como por exemplo, um recurso que possibilite ao usuário a confirmação de que uma mensagem enviada por ele foi visualizada pelo destinatário.

Cooperação (FUKS et al., 2003) é a ação em conjunto entre membros de uma equipe que compartilham o mesmo espaço de trabalho, no caso, o mesmo espaço do ambiente de colaboração, objetivando a realização de alguma tarefa em parceria. Um exemplo envolvendo manutenção colaborativa seria a realização de reuniões, através das quais há cooperação para realizar-se o planejamento das atividades de software. Uma equipe de desenvolvimento ágil, em uma dessas reuniões, delibera sobre o tempo em média que gastariam para realizar uma determinada tarefa e colaboram entre si para que se possa obter a melhor medida de tempo, utilizando, por exemplo, a técnica *Planning Poker* (SCHWABER, 2015).

Comunicação (FUKS et al., 2003) diz respeito à efetividade com a qual uma mensagem é entregue ao destinatário e, ao mesmo tempo, é entendida. Uma falha na comunicação pode fazer com que o destinatário não compreenda a intenção da comunicação estabelecida e, assim, não consiga firmar compromissos com o remetente. Um exemplo no contexto deste trabalho seria a realização de atividades nas quais há troca de informações entre membros da equipe sobre os riscos associados à tarefa de dividir um módulo que apresentam altas taxas de complexidade. Uma equipe de desenvolvimento pode no momento do planejamento, selecionar as estórias que se fazem necessárias ao desenvolvimento de um artefato de software e, por meio da comunico, definirem entre os membros da equipe de desenvolvimento e o gestor aquelas que terão maior prioridade, o tempo estimado para a conclusão ou aquelas que devem ser fragmentas para o processo de manutenção corretiva ou evolutiva do software.

Coordenação (FUKS et al., 2003) é a tarefa de articular um grupo de pessoas trabalhando em um ou vários processos que se conectam de alguma forma. Coordenar, portanto, é articular a pré-realização de uma tarefa, a realização da tarefa e a pós-realização da tarefa. Um exemplo seria a realização de atividades de planejamento de um *Sprint*[1] para equipe de desenvolvimento ágil, que envolve a seleção das estórias a serem desenvolvidas pela equipe, convidar os participantes, conduzir a reunião rumo à tomada de decisões e delegar as decisões tomadas aos responsáveis por executá-las.

---

[1] Sprint - O tempo definido dentro do qual um conjunto de atividades deve ser executado na metodologia ágil SCRUM com duração de 2 a 4 semanas.

Os elementos de colaboração apresentados podem, conforme os exemplos apresentados, beneficiar o processo de manutenção corretiva, adaptativa ou evolutiva de equipe de desenvolvimento de software.

A colaboração pode existir através do envio e recebimento de mensagens entre membros de uma equipe de diferentes formas, como exemplo, através de mensagens de correio eletrônico ou mensagens inseridas diretamente em visualizações, disponíveis no software ou no *framework* de desenvolvimento da equipe, que é o tema da próxima seção.

2.3 VISUALIZAÇÃO DE SOFTWARE

Visualização (SILVA et al., 2012) é uma importante forma de se obter compreensão, e é fundamental na tarefa de se criar um modelo mental sobre as informações. Visualizar uma dada informação pode ser útil no processo de compreensão, visando a obter conhecimento sobre o objeto analisado. A forma como as informações são disponibilizadas pode caracterizar a necessidade de se utilizar um Ambiente Interativo baseado em Múltiplas Visões (AIMV). AIMVs são ambientes que fornecem diferentes visualizações para o mesmo conjunto de informações, permitindo também que dados de interação entre usuários sejam registrados (SILVA et al., 2012). Um exemplo seria a inserção de mensagens de alerta em uma visualização, a qual permita ressaltar a Complexidade Ciclomática[2] de uma classe indicando que a mesma possui um número elevado de caminhos independentes no software. Múltiplas visões também podem evitar que interpretações errôneas possam surgir se comparadas com a análise realizada sobre um conjunto de dados visualizado apenas em uma única visão. Neste trabalho são utilizados dois paradigmas de visualização, sendo um que implementa um AIMV, ou seja, múltiplas visões para o mesmo conjunto de dados, e outro que implementa diferentes visões para diferentes conjuntos de dados.

No paradigma que implementa um AIMV, tem-se os vários tipos de visões conforme já abordado por autores, tais como (CARNEIRO et al., 2012) e (SILVA et al.,

[2] Complexidade Ciclomática - mede a quantidade de caminhos de execução independentes a partir de um código fonte dados computada por um grafo de fluxo de controle.

2012). Visualizações podem ser utilizadas também para perceber a evolução de um conjunto de dados ao longo dos anos. Um exemplo seria a visualização da contribuição dada por cada desenvolvedor em um projeto ao longo dos anos. Com uma visualização que permita acompanhar as variações do conjunto de dados isto se tornará possível (SANTANA, 2011).

2.4 ESTIMATIVA DE TEMPO

Frequentemente, estimativas de tempo em software são definidas de forma imprecisa, ocasionando atrasos na entrega das solicitações dos clientes. Uma estimativa de tempo mais correta e precisa é fundamental para a viabilidade das atividades da organização (PRIKLADNICKI e AUDY, 2007). As estimativas permitem calcular o tempo e o esforço para a entrega de uma demanda de software. Essa previsão pode auxiliar a equipe a tomar decisões ou entender a dimensão, prazo, limites e entregas possíveis para o software em desenvolvimento e manutenção (STOREY et al., 2005). Existem muitos modelos e métricas de estimativa, tais como LOC (*Lines of Code*), FP (*Function Points*), COCOMO (STOREY, 2005), bem como técnicas aplicadas ao modelo de desenvolvimento ágil, como estimativa informal, *Planning Poker, Story Points* e *Ideal Days* e PMG (Pequeno, Médio e Grande). As estimativas de projeto de software não podem ser tratadas como uma ciência exata, mas sim como uma combinação de dados históricos e técnicas sistemáticas para melhorar a precisão da estimativa (STOREY et al., 2005).

Muitas dificuldades são encontradas nesse contexto, principalmente por empresas que trabalham com metodologias ágeis, devido à imprecisão dos mecanismos de medição de esforço como *Planning Poker,* estimativas PMG, entre outros. Em sua maioria, tais técnicas usam a experiência dos desenvolvedores e sua visão sobre o tempo que será gasto para uma determinada tarefa. O julgamento individual de especialistas e abordagem de estimativa comum atualmente utilizam analogia informal como técnica primaria de estimativa (JORGENSEN, 2004). Aliada a essa informação a revisão sistemática apresentada nesse trabalho não encontrou ferramentas ou *frameworks* que utilizassem visualizações para apoiar a estimativa de esforço, reforçando a importância do presente trabalho.

Estimativas também sempre envolvem algum tipo de calibragem, seja implícita ou explícita, possibilitando assim a criação de uma estimativa baseada nos dados históricos do projeto. A razão primordial para o uso de dados históricos na realização de estimativas é a sua capacidade de gerar estimativas mais precisas. Uma maneira pela qual a subjetividade influencia negativamente as estimativas é a ideia de que um projeto novo tende a ser melhor que o anterior, influenciando assim as estimativas. Com dados históricos, se parte do princípio de que um novo projeto deve ser executado aproximadamente da mesma maneira que um anterior (MCCONNELL, 2006).

A próxima seção abordará outro tema relacionado à pesquisa deste trabalho, que são as requisições de mudanças que serviram de insumos para a análise histórica de dados realizada por essa pesquisa.

2.5 REQUISIÇÃO DE MUDANCAS

Requisições de mudanças têm como objetivo registrar a solicitação do cliente, as recomendações, custo estimado e aprovação do artefato de software para o seu desenvolvimento ou manutenção (SOMMERVILLE, 2011). Diante disso, o planejamento nas diversas fases de manutenção do software pode gerar informações quantitativas e qualitativas sobre o desempenho da equipe e do projeto em andamento como, por exemplo, o tempo médio gasto pela equipe para resolver uma requisição de mudança. Esse tempo gasto pode gerar indicadores sobre o tempo necessário para a equipe resolver uma nova requisição de que tenha as mesmas características daquelas já realizadas. Analisando historicamente o tempo médio gasto para as resoluções de requisições de mudanças, pode-se perceber o rendimento da equipe, à medida que se analisa a evolução no tempo efetivo gasto. As informações contidas nas requisições de mudanças, que possuem grau de semelhança, podem gerar indicadores para futuras manutenções de software.

A próxima seção abordará o tema Rastreabilidade visto ter sido utilizado como mecanismo na busca de dados históricos realizada por esse trabalho.

2.6 RASTREABILIDADE

A rastreabilidade de software pode ser entendida como a possibilidade de procurar artefatos que se relacionam em qualquer ponto, ao longo do ciclo de vida do software. O seu tratamento pode oferecer informações para que seja possível obter melhor entendimento sobre os relacionamentos entre os artefatos de um projeto (DALL'OGLIO, 2010).

A rastreabilidade utilizada neste trabalho permite analisar historicamente dados que possuam a mesma ideia semântica e o mesmo conjunto de alterações, permitindo assim que um conjunto de artefatos (requisições de mudanças) parecidos seja retornado pela pesquisa que utiliza essa técnica. Esses artefatos possuem uma medida de tempo gasto e estimativas para o seu desenvolvimento, os quais possibilitam gerar previsões de valores futuros baseados nos históricos passados. Os elementos retornados permitem também a geração de visões que auxiliam a equipe de desenvolvimento na tomada de decisão e estimativa de tempo.

A próxima seção abordará os trabalhos relacionados, que foram identificados por meio de uma Revisão Sistemática da Literatura, e que foram utilizados como embasamento teórico para o presente trabalho.

2.7 TRABALHOS RELACIONADOS

Com o intuito de identificar os trabalhos existentes sobre o planejamento e estimativas de tempo de atividades no contexto de Manutenção e Evolução de Software, foi realizada uma RSL - Revisão Sistemática da Literatura apresentada em MIGUEL et al. (2015). Trata-se de um método para identificar, avaliar e interpretar os trabalhos disponíveis em um contexto de pesquisa (KITCHENHAM e CHARTERS, 2007). Após a definição do protocolo do mapeamento sistemático, a *String* de busca foram definidas e executadas em bases de publicações e o retorno das buscas foram analisados. As buscas ocorreram ao longo do mês de dezembro de 2014 e posteriormente rodados em agosto de 2016. Foram encontrados, após a segunda rodada da *String* de busca, mais 138 artigos do ano de 2015 e 2016 que não estavam inseridos na primeira rodada da *String* de busca. A janela de publicações de artigos, conforme encontrado na revisão

sistemática, compreendeu o período de 2002 a 2016 salvo os artigos anteriores a essa data que são considerados seminais da área.

A *string* de busca final, utilizada na condução do processo da revisão sistemática, é descrita como segue:
(Visualization AND Evolution AND Maintenance AND Software) AND (Collaboration OR Cooperation OR Cooperative) AND (technique OR methodology OR tool OR Software OR Application OR System OR framework OR model OR Planning OR Traceability OR Requirements OR Estimative).

Como resultado de sua primeira execução, foram recuperados 692 artigos sendo que desses, 72 foram aceitos considerando-se os critérios de inclusão e de exclusão definidos no protocolo de pesquisa. Foram selecionados artigos que continham técnicas, metodologias, ferramentas ou *frameworks* que abordassem o tema de Visualização, Evolução, Manutenção Colaborativa de Software, Rastreabilidade e Estimativa. Como critérios de inclusão, foram selecionados artigos em Português e Inglês a partir de seis bases de dados digitais, disponíveis no Portal de Periódicos da CAPES, que são: IEEE, ACM, Science Direct, Engineering Village, ISI - Web of Science, Scopus.

Foram realizadas também análises adicionais, tais como as variações de publicações ao longo dos anos, os locais de publicações, os principais autores da área, e os artigos mais referenciados, bem como as ameaças à validade da pesquisa. O protocolo completo da revisão sistemática está disponível em MIGUEL et al. (2015).

Ao analisar os resultados obtidos pela RSL, foi possível identificar uma lacuna referente aos trabalhos que abordam a estimativa de tempo, especificamente no planejamento de atividades de manutenção de software. A revisão apresentou também alguns trabalhos com maior afinidade com a arquitetura proposta para o *GiveMe Effort*, os quais são descritos a seguir.

Carneiro *et al.* (2012) descrevem uma arquitetura cujo resultado é a combinação de um Ambiente Interativo baseado em Múltiplas Visões com elementos de percepção para apoiar a compreensão de software no desenvolvimento distribuído. O objetivo é fornecer informações que permitam aos integrantes das equipes conhecerem, através do ambiente, o que os demais pesquisaram, manipularam ou alteraram em um projeto de software. A partir desse objetivo, é apresentado um *framework* denominado *Collaborative SourceMiner*, integrado à IDE Eclipse, para apoiar a percepção das

atividades de compreensão no domínio da visualização de informações em um ambiente colaborativo. A ferramenta é focada na análise de código fonte, para permitir que as visualizações sejam construídas, gerando avaliações das interações dos desenvolvedores através desse código. Além disso, propõe várias visualizações que apoiam o ciclo de vida do software. Nesse contexto, a arquitetura *GiveMe Effort*, também permite a análise do código fonte. Entretanto, utiliza mecanismos de rastreabilidade para que os mesmos possam apoiar o processo de estimativa de tempo no planejamento das atividades de manutenção de software.

Storey *et al.* (2005) descrevem uma pesquisa (*survey*) sobre ferramentas de visualização existentes e propõem um *framework* conceitual que apoia o desenvolvimento e a criação de novos *framework*s. Além disso, o trabalho possibilita uma análise qualitativa da ferramenta, avaliando as características apropriadas de visualização colaborativa de software. Por se tratar de um *framework*, não há uma implementação que possibilite a análise automatizada de outros *framework*s, ou a análise de código fonte, tampouco que analise o histórico de mudanças em repositórios de mudanças. Adicionalmente, não apoia o processo de planejamento de atividades de manutenção de software num ambiente de manutenção colaborativa.

Churrasco (D'AMBROS, 2003) é uma ferramenta para apoiar a análise da evolução colaborativa de software por meio de uma interface Web. Destacam-se como pontos positivos da ferramenta a análise do código fonte do repositório SVN (PILATO, 2008) e também do repositório de defeitos Bugzilla (SERRANO, 2005). Possui ainda uma interface com recursos, tais como anotação colaborativa e visualização gráfica de processo de manutenção e de métricas associadas. Oferece o suporte de visualizações para que os usuários, de forma colaborativa, interajam durante o processo de manutenção e evolução de software. Churrasco possui a capacidade de obter, automaticamente, dados históricos de repositórios de código fonte (SVN) e a capacidade de apoiar equipes distribuídas. Adicionalmente, apoia a colaboração entre membros, em atividades de compreensão sobre a evolução do software. Entretanto, não apoia a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis e em um ambiente distribuído de software.

FRASR (*Framework for Analyzing Software Repositories*) (PONCIN, 2011) é um *framework* que permite a análise de processos de desenvolvimento de software armazenados em repositórios de dados. Dentre suas características destaca-se a capacidade de extrair informações de sistemas de controle de versão e repositórios de defeitos. No contexto de repositório de defeitos, a solução proposta nesta pesquisa também é capaz de extrair dados sobre defeitos, como por exemplo, o tipo de defeito (funcionalidade, usabilidade, desempenho, saída), e informações para os cálculos das estimativas de tempo, conforme será abordado na seção 3, que explica o funcionamento da arquitetura. No entanto, não apoia a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis e em um ambiente distribuído de software.

Kevic et al. (2013) descrevem uma ferramenta que permite a análise de defeitos, com o objetivo de encontrar similaridades entre eles. Foi desenvolvida para ser utilizada de forma colaborativa, através da qual, desenvolvedores interagem e são capazes de encontrar similaridades entre defeitos, em relação ao que está sendo desenvolvido. Primeiramente, um novo defeito é cadastrado e aguarda até que ele possa ser resolvido. Nesse momento um conjunto de desenvolvedores analisarão os defeitos que já tenham sido resolvidos anteriormente, e que sejam similares ao defeito em questão. As similaridades são identificadas a partir de um vetor de consulta, baseado nas palavras encontradas no repositório de defeitos, e que são indexadas pelas palavras encontradas e os defeitos relacionados às mesmas. Para gerar uma lista ordenada dos relatórios de defeitos semelhantes, a ferramenta calcula a similaridade entre o vetor de consulta e o vetor de cada documento. A ferramenta utiliza os valores mais altos em relação ao documento analisado para indicar um conjunto de requisições de defeitos com similaridades. Em seguida, podem-se observar as mudanças ocorridas, em termos de código fonte, indicando que o defeito a ser resolvido poderá impactar aqueles mesmos trechos de código, dada a similaridade entre eles. A ferramenta atua considerando dados históricos tanto de defeitos como de alterações em código fonte. Possui ainda a capacidade de apoiar a colaboração entre equipes de manutenção. Similarmente, a solução proposta nesta pesquisa possibilita a análise de defeitos de forma histórica, com o objetivo de descobrir os defeitos e os trechos de código fonte que foram impactados em cada solicitação de manutenção do software. Entretanto, esta ferramenta, assim

como as demais apresentadas, não apoia a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis e em um ambiente distribuído de software.

Tracer (MOHAN, 2008) é uma ferramenta que suporta a criação de uma rede de rastreabilidade, representando a associação entre diversos componentes, tais como requisitos e modelos de projeto. É integrada ao MS Visual SourceSafe para o controle de versão. Permite geração de rastreabilidade em um nível de granularidade mais baixo, relacionando, por exemplo, um requisito específico a uma classe, ou uma requisição de mudança a um diagrama UML (*Unified Modeling Language*). Assim como a ferramenta Tracer (MOHAN, 2008), a solução proposta também utiliza técnicas de rastreabilidade, objetivando descobrir métodos, classes, com marcadores em repositórios de defeitos, e requisições de mudança com características similares, mas, assim como as demais apresentadas, não apoia a estimativa de esforço em um AIMV, em projetos ágeis e em um ambiente distribuído de software.

A Tabela 1 sintetiza as soluções existentes, previamente apresentadas, e as relaciona com um conjunto de critérios de análise. A escolha desses critérios se baseia nas condições necessárias para a estruturação da solução apresentada na seção 3. Os critérios são: (i) Análise de código fonte, cujo objetivo é descrever se a ferramenta analisa o código fonte de aplicações e apoia a manutenção ou evolução de software; (ii) Análise de repositório de requisições de mudança, cujo objetivo é descrever se a ferramenta oferece suporte à análise de repositório de requisições de mudança para apoiar a manutenção e evolução de software; (iii) IDE (*Integrated Development Environment*), cujo objetivo é descrever a IDE em que a ferramenta se baseia para seu funcionamento e exibição das visualizações propostas; (iv) Paradigma, que tem como objetivo descrever o paradigma de desenvolvimento de software (Orientado a objetos, Funcional, entre outros) através do qual o *framework* ou ferramenta é estruturado; (v) Modelo 3C (PIMENTEL et al., 2008), que objetiva descrever se o *framework* ou ferramenta oferece suporte aos elementos de colaboração, tais como: comunicação, cooperação e coordenação, que são pertinentes à manutenção colaborativa de software; (vi) Rastreabilidade, que objetiva identificar se o *framework* ou ferramenta utiliza técnicas de rastreabilidade de código fonte ou de outros tipos de repositórios; (vii)

Análise de planejamento, que objetiva analisar se o *framework* ou ferramenta possui algum tipo de mecanismo de planejamento das atividades de manutenção e evolução de software.

Tabela 1: Tabela Comparativa entras as Ferramentas/*Frameworks*

						3C				
Ferramenta/ Framework	Código Fonte	Repositório de Defeitos	Repositório de Processos	IDE	Paradigma	Comunicação	Coordenação	Cooperação	Rastreabilidade	**Planejamento**
Collaborative SourceMiner (CARNEIRO, 2012)	Sim	Não	Não	Eclipse	OO	Sim	Sim	Sim	**Não**	**Não**
Storey (STOREY, 2005)	Não	Não	Não	Conceitual	Conceitual	Não	Não	Não	**Não**	**Não**
Churrasco (D'AMBROS, 2003)	Sim	Sim	Não	Browser (IE, Firefox, Chrome)	OO	Sim	Sim	Sim	**Sim**	**Não**
FRASR (PONCIN, 2011)	Não	Sim	Não	Própria	Ambos	Não	Não	Não	**Não**	**Não**
Bug management (KEVIC, 2013)	Sim	Sim	Não	Própria	Ambos	Sim	Sim	Sim	**Sim**	**Não**
Tracer (MOHAN, 2008)	Sim	Não	Sim	Própria	OO	Não	Não	Não	**Sim**	**Não**

Conforme apresentado na Tabela 1, os *framework*s e ferramentas apresentados reforçam a importância do objetivo deste trabalho, visto que nenhum deles apoia algum tipo de técnica para a Estimativa de Esforço no planejamento de atividades de manutenção de software aliados a um AIMV, no contexto de projetos ágeis distribuídos de software. Além disso, nem todas as ferramentas ou *frameworks* apresentados apoiaram a compreensão e visualização ao longo do ciclo de vida do software.

Alguns requisitos não funcionais, ficaram evidentes em alguns trabalhos relacionados, mas, não foram encontrados nas pesquisas realizadas um trabalho que abordasse o conjunto de requisitos propostos ao projeto arquitetural do *framework* proposto. A portabilidade e reuso são abordados pelos trabalhos Collaborative SourceMiner (CARNEIRO, 2012) e Churrasco (D'AMBROS, 2003). A interoperabilidade não foi abordada de forma completa em nenhuma das soluções apresentadas. A extensibilidade é considerada no trabalho Churrasco (D'AMBROS, 2003) e é proposta no decorrer do trabalho apresentado pela ferramenta. Pode-se notar que, de uma forma geral, todos os requisitos em conjunto não são abordados pelos trabalhos apresentados. Entretanto, a forma como algumas soluções abordam seus requisitos forneceu, para o presente trabalho, o auxílio na criação da arquitetura e na utilização dos requisitos na construção do *framework*.

2.8 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

Este capítulo apresentou os conceitos base deste trabalho. Uma visão geral das relações existentes entre os respectivos conceitos e o problema geral abordado neste trabalho foi mostrada, criando um cenário para a proposta da solução que combina os conceitos desta pesquisa. Foram apresentados também, os trabalhos relacionados, selecionados a partir de uma revisão sistemática da literatura, os quais reforçam a relevância deste trabalho. A revisão sistemática apontou trabalhos relevantes para a área e que puderam nortear o desenvolvimento do trabalho proposto no próximo capítulo. Pôde-se observar que nem todos os trabalhos encontrados possuíam todas as características propostas pelo presente trabalho, ou seja, uma abordagem que viabilize a estimativa de esforço para as equipes de desenvolvimento, provendo elementos de visualização, em um ambiente integrado às atividades de compreensão e manutenção de software em um AIMV.

A próxima seção abordará a solução proposta por esse trabalho bem como a arquitetura e o funcionamento de seus componentes e a integração dos mesmos com a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis distribuídos de software.

3. GIVEME EFFORT

A solução para o problema destacado se relaciona ao desenvolvimento de um *framework* que integra técnicas de estimativa de esforço no contexto de manutenção e compreensão de software em um AIMV, no contexto de projetos ágeis e em um ambiente distribuído de software.

GiveMe Effort, é um *framework* baseado em múltiplas visões interativas para apoiar a manutenção e compreensão do software e está integrado a ambientes de compreensão de software interativos baseados em múltiplas visões (AIMVs). Busca também apoiar a colaboração entre equipes geograficamente distribuídas, nas tarefas de manutenção e evolução de software. Um ambiente baseado em AIMVs (SILVA et al., 2012) possibilita a análise de dados, em diferentes perspectivas, através da combinação de visualizações distintas, as quais podem ser devidamente coordenadas.

O *GiveMe Effort* foi projetado para ser integrada à *GiveMe Infra* (TAVARES, 2015) que é um produto desenvolvido no Núcleo de Pesquisas em Engenharia do Conhecimento da UFJF (NEnC), sendo o *framework* proposto neste trabalho, parte integrante dessa infraestrutura.

Uma entrada para a arquitetura do *GiveMe Infra* é a indicação de um módulo ou componente que se deseja dar manutenção. Em seguida, os dados históricos referentes ao projeto são importados e filtrados para uma análise estatística, bem como para o cálculo das métricas disponíveis, as quais são descritas na sequência:

- Tamanho: número de classes, número de métodos por classe, número de subsistemas, número de linhas de código fonte;
- Periodicidade: intervalo de tempo entre versões de um projeto;
- Complexidade: profundidade de herança por classe, número de filhos por classe, acoplamento entre objetos, falta de coesão entre métodos e complexidade Ciclomática.

A informação gerada é então persistida e disponibilizada para exportação ou para acesso de visualizações em outras ferramentas, via provedor, ou seja, um único ponto pelo qual visualizações e ferramentas que queiram se integrar ao *GiveMe Infra* podem obter informações sobre os dados históricos processados.

Entretanto, a *GiveMe Infra* não apoia a estimativa de esforço nas atividades de manutenção de software. A atividade de estimativa de esforço visa calcular o tempo e planejamento das manutenções corretivas, adaptativas ou evolutivas. A solução proposta neste artigo integra-se a essa ferramenta, possibilitando assim, a utilização de dados históricos de softwares para auxiliar no planejamento de manutenções corretivas e adaptativas através dos recursos oferecidas pela *GiveMe Infra*.

Além disso, a proposta da solução disponibiliza visualizações e acompanhamentos das atividades de estimativas de tempo e de planejamento. A arquitetura em módulos do *framework,* possibilita a customização das visualizações e das métricas suportadas pela *GiveMe Effort*, e a integração com diferentes ferramentas.

A arquitetura proposta para o *framework GiveMe Effort*, objetiva a geração das visualizações, o apoio à gestão e controle da Estimativa de Esforço de atividades de evolução de software, baseado no conjunto histórico de dados de manutenção analisados pela ferramenta *GiveMe Infra* (TAVARES, 2015). A Figura 2, apresenta a arquitetura de componentes do *GiveMe Infra*, bem como, em destaque a integração da solução proposta nesse trabalho. Os componentes apresentados na cor azul fornecem mecanismos para a integração com a arquitetura proposta nesse trabalho, como o *GiveMe Trace e GiveMe Repository,* que fornecem respectivamente auxílio à rastreabilidade e acesso ao repositório de dados. Os componentes em rosa, fornecem mecanismos de visualização que apoiam o *GiveMe Infra* na disponibilização de informações de manutenção e evolução do software. Por fim, o componente destacado em vermelho representa a ligação do presente trabalho com a infraestrutura do GiveMe Infra. O detalhamento completo do funcionamento dos componentes foi realizado por TAVARES (2015).

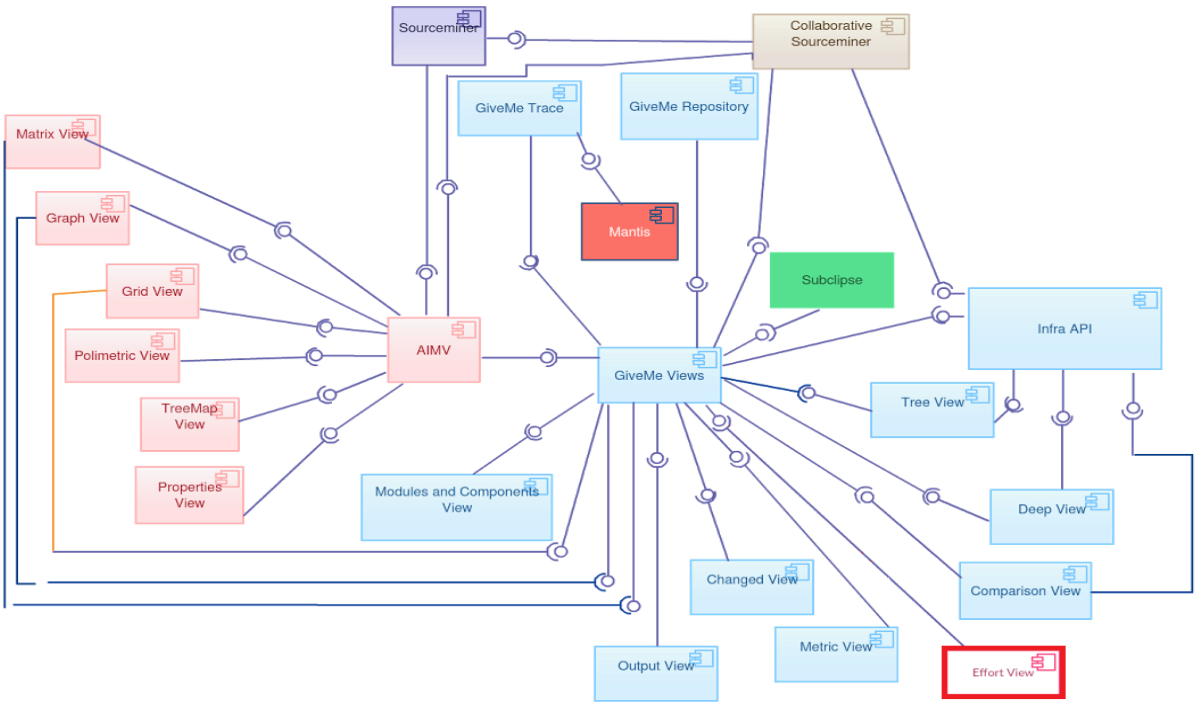


Figura 2: Visão geral do diagrama de componentes do *GiveMe Infra* e a integração da solução atual a infraestrutura (TAVARES, 2015)

Alguns requisitos não funcionais merecem destaque para a geração do modelo arquitetural do framework, tais como: (i) a portabilidade, foi considerada para que o sistema fosse implementado de forma a funcionar com diversos sistemas operacionais (Windows, Linux, MacOS), (ii) o reuso, possibilitará a utilização dos módulos da ferramenta no futuro; (iii) a usabilidade, permite que a ferramenta funcione de forma fácil pelos usuários; (iii) a interoperabilidade, deve possibilitar a troca de dados com a *GiveMe Infra*; (iv) a extensibilidade, foi considerada como estratégia para permitir o suporte as atualizações e novas versões do *GiveMe Infra* e dos módulos do *framework GiveMe Effort*.

Como requisitos funcionais na arquitetura do framework pode-se destacar: (i) prover **múltiplas técnicas** para o cálculo da estimativa de esforço, (ii) prover **múltiplas visualizações** para auxiliar na compreensão do código, (iii) permitir a **integração** com outros *frameworks* como o *GiveMe Infra*, (iv) possibilitar a **modularidade** das técnicas permitindo sua substituição.

Na subseção seguinte, é apresentada a arquitetura *GiveMe Effort* considerando, sobretudo, que deve ser aderente ao *framework GiveMe Infra*, e atender os requisitos funcionais e não funcionais apresentados, bem como fornecer mecanismos para a visualização e planejamento da estimativa de esforço ao longo do ciclo de vida do software.

3.1 ARQUITETURA

A Figura 3 apresenta a arquitetura do *GiveMe Effort*. Baseado nos conceitos apresentados na seção 2 desse trabalho, foi desenvolvido o *framework* de estimativa de esforço, com suas visualizações e as técnicas utilizadas para a construção da estimativa, bem como a sua integração à infraestrutura do *GiveMe Infra* (TAVARES, 2015).

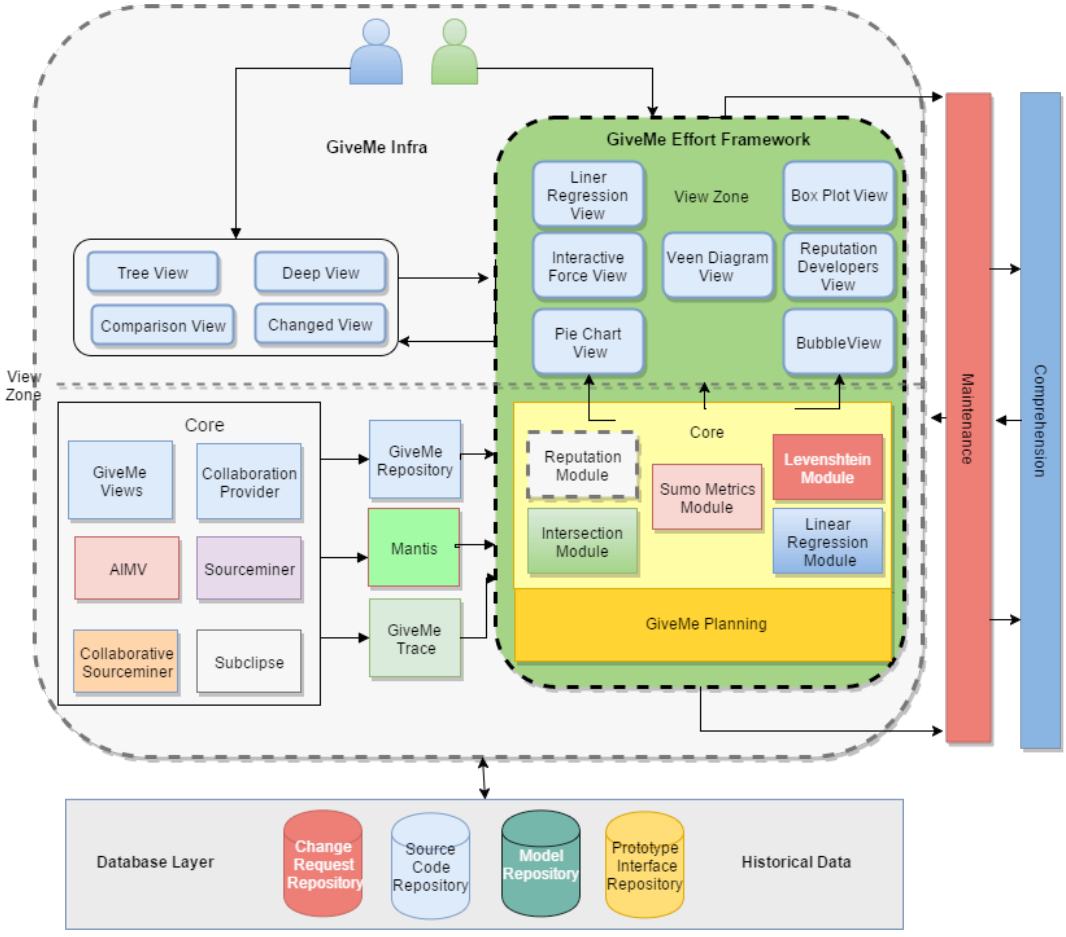


Figura 3: Visão geral da arquitetura do *GiveMe Effort*

A View Zone da infraestrutura do GiveMe Infra, apresenta visualizações como Tree View, Deep View, Comparation View e Changed View que estão detalhadas no

trabalho proposto por TAVARES (2015). Essas visualizações permitem ao usuário a partir do repositório de dados de código fonte obter informações visuais a respeito da manutenção e evolução de software. A infraestrutura proposta para o GiveMe Effort, permite, a partir de uma requisição de mudança a ser analisada, obter informações adicionais a respeito do trecho de código fonte modificado historicamente, permitindo ao usuário do *framework* tomar decisões a respeito do tempo e complexidade de uma manutenção corretiva ou adaptativa de software.

O núcleo do GiveMe Infra permite processar e gerar informações para as visualizações apresentadas. Serve como um processador de dados da base histórica que permite a geração de dados para outros componentes utilizados pela infraestrutura do GiveMe Effort, como o componente Mantis. Este componente integra os artefatos e código fonte utilizado propondo mecanismos para que a rastreabilidade dos dados (GiveMe Trace) possa encontrar requisições no repositório de dados (GiveMe Repository), bem como os tempos gastos para a estimativa do GiveMe Effort.

A Figura 3 apresenta a camada referente à base de dados, que é composta por 4 repositórios. O repositório *Change Request Repository* possui as requisições de mudanças do software ao longo do tempo, as quais são utilizadas pelo núcleo do *framework* para a busca de informações históricas. Essas informações servem de comparação e base para o cálculo da estimativa de esforço. O *Source Code Repository* é o repositório de código fonte, tal como GIT (2015) ou SVN (2015), possui dados históricos e evolutivos do software. Ainda, é possível coletar informações históricas do *Model Repository*, o qual se refere ao repositório de modelo de dados, com as documentações existentes do software armazenadas de forma histórica. Serve como base para o planejamento de esforço. Por fim, o *Prototype Interface Repository* o qual possui protótipos de interface armazenados de forma histórica. Possibilita a análise de similaridade entre o protótipo construído para a nova funcionalidade, e modelos pré-existentes de interface.

A partir desses dados, é possível descobrir os principais componentes da aplicação, a concentração de alterações no sistema, os *Sprints* utilizados e os códigos fonte relacionados às solicitações de mudança em manutenção de software. Pode-se também prover, a partir desses dados, um cenário real a respeito da aplicação e seu ciclo de vida, baseado em suas mudanças. O repositório *Mantis* (Mantis, 2015) possui registros de defeitos e solicitações de melhorias reportadas pelos usuários, analistas de

suporte ou desenvolvedores de software, ao longo do ciclo de vida do sistema. Esses dados históricos, aliados às mudanças de código fonte, podem traçar panoramas importantes a respeito do software, tais como suas tendências futuras de manutenções corretivas ou evolutivas. Dados de uma requisição, gerados pelas ferramentas, podem ser associados a partes do código fonte em que foram feitas as alterações. Portanto, pode-se obter, por meio de informações sobre rastreabilidade, o problema que levou a alterações no código fonte, e as partes do código fonte que foram alteradas em função de uma requisição de mudança reportada. Os cruzamentos podem gerar uma gama de informações para futuras tomadas de decisão.

A arquitetura do *framework GiveMe Effort,* destacado na Figura 3, apoia o planejamento das atividades de software através da sugestão de estimativas (*GiveMe Planning*). Após o processamento dos dados pelos respectivos repositórios, é possível estimar o esforço das atividades, por meio do registro histórico dos dados extraídos das requisições de mudança e do código fonte, desde que estes possuam similaridades com atividades anteriormente realizadas.

Através da *View Zone,* é possível obter visualizações das atividades de planejamento de esforço que podem ser úteis no processo de tomada de decisão. O núcleo do *framework* possui vários módulos que auxiliam os cálculos de estimativa, os quais viabilizam a geração de uma estimativa de esforço nas atividades de manutenção (corretivas, adaptativas e evolutivas) de um software.

O *framework GiveMe Effort* utiliza a ferramenta *GiveMe Trace* (LÉLIS, 2014), para o suporte à rastreabilidade. A ferramenta foi desenvolvida no Núcleo de Pesquisas em Engenharia do Conhecimento da UFJF (NEnC), utiliza técnicas de rastreabilidade de código fonte entre o número da versão e as classes ou métodos alterados no código fonte. A *GiveMe Trace* opera sobre linhas de código fonte, buscando métodos e classes utilizados na manutenção de software, diferentemente de outras abordagens que consideram a rastreabilidade entre arquivos, tornando possível rastrear, a partir de uma requisição de mudança informada, os códigos fontes relacionados a ela. Esse rastreamento apoia o processo de estimativa de tempo em atividades de manutenção do software. A ferramenta *GiveMe Trace* é parte integrante da infraestrutura *GiveMe Infra* (TAVARES, 2015), utilizada pelo *framework GiveMe Effort* e oferece suporte à evolução de software baseada em múltiplas visões.

A subseção seguinte detalha o funcionamento de cada um dos componentes apresentados na infraestrutura do *GiveMe Effort,* bem como sua contribuição para o processo de estimativa de esforço.

3.2 NÚCLEO – SUMO METRICS MODULE

O *Sumo Metrics Module* é um módulo para cálculo de similaridade textual entre a requisição de mudança solicitada e o histórico de dados coletado. A respectiva técnica destaca-se pela característica de comparação de *strings* que, mesmo grafadas de forma, ligeiramente, diferente, transmitem a mesma ideia ou informação. A Sumo-metric (CORDEIRO, 2007), busca estabelecer a noção de ligações lexicais, exclusivas em um par de frases (*strings*), e definir o grau de relevância entre as mesmas. O número de ligações entre as duas sentenças é definido como λ e a quantidade de palavras na maior e menor sentença, como x e y, respectivamente. As frações $\frac{\lambda}{x}$ e $\frac{\lambda}{y}$ são valores no intervalo [0, 1], indicando conectividade lexical normalizada entre as frases. A equação descrita a seguir, primeiramente busca a resolução da função S (x, y, λ), onde α, β ∈ [0, 1] e α + β = 1.

$$S(x, y, \lambda) = \alpha \log_2(\frac{x}{\lambda}) + \beta \log_2(\frac{y}{\lambda})$$

Na sequência é aplicada a segunda equação, conforme apresentado a seguir.

$$S(S_a, S_b) = \begin{cases} S(x, y, \lambda) & if \;\; S(x, y, \lambda) < 1.0 \\ e^{-k * S(x,y,\lambda)} & Caso\ Contrário \end{cases}$$

Os parâmetros α e β representam o peso dos dois componentes principais envolvidos no cálculo, a partir de uma interpolação linear. O efeito da utilização da segunda equação é penalizar gradualmente pares que não sejam semelhantes, valorizando os pares realmente similares.

Essa busca por similaridades permite encontrar requisições de mudança que possuem maior afinidade com as requisições em manutenção, possibilitando assim, o cálculo do esforço estimado de resolução. Este cálculo é baseado nos tempos consumidos nas requisições de mudança similares.

3.3 NÚCLEO – LEVENSHTEIN MODULE

O *Levenshtein Module* é outro módulo para apoiar o cálculo de similaridade textual entre a requisição solicitada e a base histórica existente. À distância Levenshtein (LEVENSHTEIN, 1966), ou distância de edição entre duas *strings* (duas sequências de caracteres), é dada pelo número mínimo de operações necessárias para transformar uma *string* em outra, com operações de edição, inserção, eliminação ou substituição de caracteres.

Para efetuar o cálculo e obter o número de trocas entre as *strings*, utiliza-se o algoritmo *bottom-up*, de programação dinâmica, com uma matriz $(n + 1) \times (m + 1)$, na qual n e m representam o número de caracteres das duas *strings*.

A matriz inicializada mede (m, n), correspondente à distância Levenshtein entre o prefixo de m-caracteres com o prefixo n-caracteres da outra palavra. A matriz pode ser preenchida a partir do canto superior esquerdo para o canto inferior direito. Cada salto horizontal ou vertical corresponde a uma entrada ou uma exclusão, respectivamente. O custo é normalmente definido como 1 (um) para cada uma das operações. O salto diagonal pode custar 1 ou 0, caso os dois caracteres na linha e coluna não sejam correspondentes. Cada célula sempre minimiza o custo localmente. Sendo assim, o número no canto inferior direito é a distância Levenshtein entre ambas as palavras. A Figura 5 mostra um exemplo que apresenta a comparação entre as palavras "Meilenstein" e "levenshtein".

		m	e	i	l	e	n	s	t	e	i	n
	0	1	2	3	4	5	6	7	8	9	10	11
l	1	1	2	3	3	4	5	6	7	8	9	10
e	2	2	1	2	3	3	4	5	6	7	8	9
v	3	3	2	2	3	4	4	5	6	7	8	9
e	4	4	3	3	3	3	4	5	6	6	7	8
n	5	5	4	4	4	4	3	4	5	6	7	7
s	6	6	5	5	5	5	4	3	4	5	6	7
h	7	7	6	6	6	6	5	4	4	5	6	7
t	8	8	7	7	7	7	6	5	4	5	6	7
e	9	9	8	8	8	7	7	6	5	4	5	6
i	10	10	9	8	9	8	8	7	6	5	4	5
n	11	11	10	9	9	9	8	8	7	6	5	4

Figura 4: Exemplo de aplicação da distância de Levenshtein

Existem dois caminhos possíveis através da matriz, que, na verdade, produzem a solução de menor custo. A Figura 6 demonstra as operações realizadas para o cálculo da distância, onde o sinal de igualdade (=) indica que não há necessidade de substituição de caracteres. O sinal "o" indica substituição de um caracter. Já o sinal "+" indica a inclusão de um novo caracter. Por fim o sinal "-" indica a eliminação de um caracter.

Figura 5: Resultado da aplicação da distância de Levenshtein

Essa busca pela quantidade de alterações entre as *strings* permite encontrar requisições de mudança que possuem maior afinidade (similaridade) com a requisições em manutenção, permitindo assim, o cálculo do esforço estimado de resolução baseado nos tempos consumidos nas requisições de mudança similares.

3.4 NÚCLEO – REPUTATION MODULE

O componente *Reputation Module* é o módulo de cálculo de reputação dos desenvolvedores. A reputação foi considerada para esse trabalho visto que, após analisar a base de dados utilizada na avaliação e desenvolvimento da solução, percebeu-se que se possuía não só o nome dos desenvolvedores, mas também indicadores como tempo em um projeto e quantidade de requisições de mudanças atribuídas e resolvidas por ele. Mediante isso, criou-se um módulo para o cálculo da reputação dos desenvolvedores permitindo, como trabalhos futuros, a inclusão de novas técnicas de cálculo de reputação para auxiliar na tomada de decisão e Estimativa de Esforço.

Esse módulo considera o índice de reputação para cálculo do total de requisições de mudanças, realizadas pelo usuário ao longo do tempo. O índice de cada desenvolvedor é normalizado em função dos demais desenvolvedores, utilizando a Norma Euclidiana (FRANCO, 2006) de um vetor para que os valores estejam em um índice com base 0 a 1. A norma euclidiana, que traduz o conceito habitual de comprimento de um vetor a partir de sua origem, tem a seguinte expressão:

$$\left[\underset{n}{\rightarrow}\right] = \sqrt{\underset{v}{\rightarrow} . \underset{v}{\rightarrow}} = \sqrt{{v_1}^2 + {v_2}^2 + {v_n}^2}$$

Ou pode ser dado pela formula abaixo, onde x é um vetor em Rn e Xi é o valor do vetor no eixo i.

$$\| \mathbf{x} \|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

A fórmula acima retorna o comprimento Euclidiano do vetor $\underset{n}{\rightarrow}$. Logo após esse cálculo, normalizam-se os valores $\underset{u}{\rightarrow}$ em função da norma Euclidiana encontrada $\underset{n}{\rightarrow}$. Conforme a expressão abaixo.

$$\underset{u}{\rightarrow} = \frac{\underset{v1}{\rightarrow}}{\left[\underset{n}{\rightarrow}\right]} \; .. \; \frac{\underset{vn}{\rightarrow}}{\left[\underset{n}{\rightarrow}\right]}$$

O resultado final é um valor para cada índice de reputação do vetor em função da reta, conforme Figura 6.

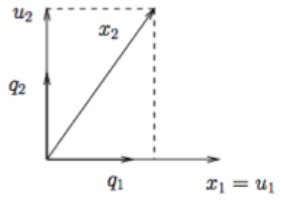


Figura 6: Norma Euclidiana de um vetor

O índice de reputação permite ranquear os melhores desenvolvedores em cada uma das requisições de mudanças e assim selecionar apenas as requisições em que esse índice é maior, ou seja, os desenvolvedores com maior índice possuem uma maior confiança e, portanto, uma maior assertividade no tempo de desenvolvimento. Dessa forma, podem-se definir as requisições que devem ser desenvolvidas por desenvolvedores específicos da equipe.

3.5 NÚCLEO - INTERSECTION MODULE

O componente foi desenvolvido para capturar as interseções entre as técnicas. Sua criação se baseou na observação empírica de que muitas das técnicas já apresentadas, possuíam em um dado momento requisições de mudanças em comum.

O componente *Intersection Module* é o módulo responsável por encontrar a interseção entre as três métricas calculadas (Sumo, Levenshtein e Reputação). A interseção (SHIN, 1994) é um conjunto de elementos que pertencem, simultaneamente, a dois ou mais conjuntos. A utilização de uma técnica para encontrar a interseção objetiva a identificação das requisições de mudanças que atendem, simultaneamente, as três técnicas descritas anteriormente (Sumo, Levenshtein e Reputação). Dessa forma, é possível encontrar elementos que possuem uma maior afinidade, aumentando a probabilidade de que sejam semelhantes em suas descrições.

A Figura 7 apresenta um exemplo de utilização dessa técnica. Pode-se notar que no circulo verde temos 369 requisições de mudança selecionadas pela técnica Reputation, no circulo azul 14 requisições com a técnica Sumo e no circulo laranja da técnica Levenshtein, 1452 requisições de mudança foram encontradas. A Interseção entre as técnicas encontrou 5 (cinco) requisições de mudança que estão presentes em todas as 3 (três) técnicas apresentadas.

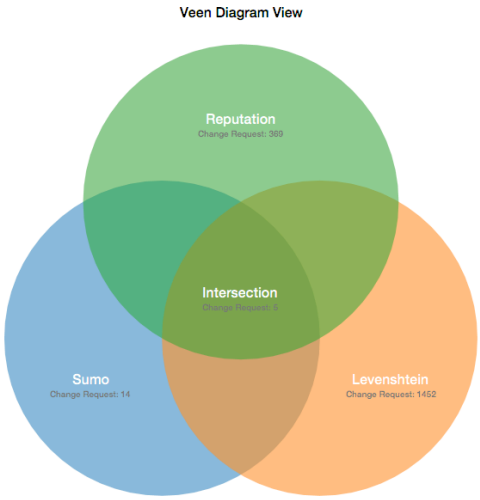


Figura 7: Interseção entre conjuntos nas técnicas apresentadas pelo *framework*

3.6 NÚCLEO – LINEAR REGRESSION MODULE

O módulo de regressão Linear foi adicionado ao *framework* a partir da análise das técnicas atuais para se fazer estimativas quantitativas de valores, que foram retornadas na Revisão Sistemática da Literatura. Essa técnica é utilizada extensivamente pelas áreas matemática e estatística há anos, e possui uma definição formal bem elaborada,

bem como o descritivo de seu funcionamento o que possibilitou a criação do algoritmo para o seu cálculo. Nas observações empíricas dos resultados encontrados pelas outras técnicas, analisou-se que, visto que possuíamos na base de dados o tempo e os valores históricos das requisições de mudanças que se assemelhavam à requisição solicitada para ser analisada pelo *framework*, poderíamos propor um próximo valor utilizado a técnica de Regressão Linear (FEDOTOVA, 2013).

O *Linear Regression Module* calcula a Regressão Linear (FEDOTOVA, 2013), por meio de uma equação para se estimar a condicional (valor esperado) de uma variável y, dados os valores da variável x. A Regressão Linear trata da questão de se estimar um valor condicional não esperado, é chamada linear porque busca considerar que a relação da resposta às variáveis é uma função linear de alguns parâmetros. A análise de regressão (MONTGOMERY, 2012) consiste na realização de uma análise estatística, com o objetivo de verificar a existência de uma relação funcional entre uma variável dependente (y) e a variável independente (x), a fim de tentar estabelecer uma equação que representa o fenômeno de regressão em estudo, pode-se utilizar o gráfico chamado de Diagrama de Dispersão, conforme demonstrado na Figura 8, para verificar como se comportam os valores da variável dependente (y) em função da variação da variável independente (x).

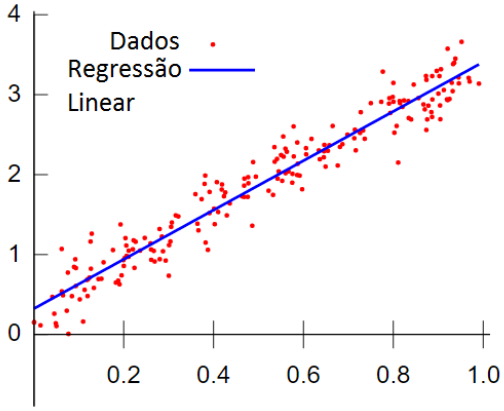


Figura 8: Regressão Linear – Representação Gráfica (MONTGOMERY, 2012)

Para obter-se a relação funcional (MONTGOMERY, 2012), é necessário basear-se na obtenção de uma equação estimada, sendo que a mesma deve demonstrar as distâncias entre os pontos do diagrama e os pontos da curva do modelo matemático e que, em relação a todos os pontos, sejam os menores possíveis. Este método é denominado de Método dos Mínimos Quadrados - LSM (*Least Squares Method*)

(MONTGOMERY, 2012). Nesse método a soma de quadrados das distâncias entre os pontos do diagrama e os respectivos pontos na curva da equação estimada é minimizada, obtendo-se uma relação funcional entre x e y, com um mínimo de erro possível.

O modelo estatístico é representado pela equação:

$$Y_i = \beta_0 + \beta_1 + X_i + e_i$$

Onde: Y é o valor observado para a variável dependente Y no i-ésimo nível da variável independente X. β_0 é a constante de regressão, representa o intercepto da reta com o eixo dos Y. β_1 é o coeficiente de regressão, o qual representa a variação de Y em função da variação de uma unidade da variável X. X_i é *o* i-ésimo nível da variável independente X (i = 1,2,...,n). O valor e_i é o erro que está associado à distância entre o valor observado Yi e o correspondente ponto na curva, para o mesmo nível i de X. Para obter a equação estimada, utiliza-se o LSM, visando à minimização dos erros. Assim, tem-se a seguinte equação:

$$e_i = \mathrm{Y_i} - \beta_0 - \beta_1 X_i$$

Elevando ambos os membros da equação ao quadrado, tem-se:

$$e_i^2 = \left[\mathrm{Y_i} - \beta_0 - \beta_1 X_i\right]^2$$

Aplicando o somatório, tem-se:

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left[\mathrm{Y_i} - \beta_0 - \beta_1 X_i\right]^2$$

Por meio da obtenção dos valores estimados de β_0 e β_1, é possível alcançar a minimização da soma de quadrados dos erros. Para se encontrar o mínimo para uma equação, deve-se derivá-la em relação à variável de interesse e igualá-la a zero. Derivando a primeira expressão em relação a β_0 e β_1 e igualando-as a zero, poderemos obter duas equações que, juntas, vão compor o sistema de equações normais. A solução desse sistema será:

$$\hat{\beta}_1 = \frac{\sum x_i y_i - \frac{\sum x_i \sum y_i}{n}}{\sum x_i^2 - \frac{\left(\sum x_i\right)^2}{n}} = \frac{SPD_{xy}}{SQD_x} \quad \text{e} \quad \hat{\beta}_0 = \overline{Y} - \hat{\beta}_1 \overline{X}$$

Uma vez obtidas estas estimativas, podemos escrever a equação estimada como:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$$

A Regressão Linear (FEDOTOVA, 2013), permite, a partir dos tempos obtidos pelos esforços das técnicas Sumo Levenshtein e Reputação, prever o esforço para uma nova requisição de mudança. Esse método apresentou resultados interessantes na geração de valores estimados, conforme será apresentado na seção 4.

3.7 VIEW ZONE

A *View Zone* apresenta sete visualizações e acompanhamentos das atividades de estimativas de esforço e de planejamento, as quais auxiliam a manutenção e compreensão do software.

As visualizações foram implementadas em JavaScript utilizando como base alguns *frameworks* conhecidos do mercado como Highcharts[3] e D3plus[4]. Estes *frameworks* foram alterados e ajustados para que fossem aderentes às visualizações propostas pelo GiveMe Effort. A escolha dos gráficos apresentados, se deu em função da natureza proposta pelo presente trabalho e foram criadas de forma incremental à medida que se percebia a necessidade de mais informações visuais à respeito dos dados encontrados pelas técnicas de estimativa. Procurou-se ao longo do desenvolvimento das visualizações, enriquecê-las com informações adicionais relacionadas à origem dos dados e como os mesmos poderiam se integrar à estimativa de esforço. As visualizações propostas pelos *toolkits* apresentados, promovem integração com os dados e a estimativa de esforço calculada. É possível também promover uma interação entre as diversas visualizações propostas, e abertas simultaneamente. As subseções seguintes detalham cada uma das visualizações, bem como, as suas utilidades e as informações adicionais incluídas em cada uma delas.

---

[3] Highcharts - Disponível em: http://www.highcharts.com. Acesso em 01 ago. 2016.

[4] D3plus- Disponível em: http://www.d3plus.org. Acesso em 01 ago. 2016.

3.8 PIE CHART VIEW

A visualização *Pie Chart View*, disposta na Figura 9, permite representar, de forma visual, a quantidade de requisições de mudanças selecionadas pelo *framework* em cada uma das técnicas apresentadas. Essa informação é útil na rápida identificação, de forma visual, de qual técnica está selecionando o maior número de requisições e como está a proporcionalidade de uma determinada técnica em relação às demais. Essa visualização foi selecionada devido à necessidade de termos de forma visual os totalizadores de cada uma das técnicas, podendo-se perceber quais técnicas encontram mais requisições de mudanças em relação ao total. Como resultado, mediante os valores apresentados o usuário pode recalibrar a ferramenta.

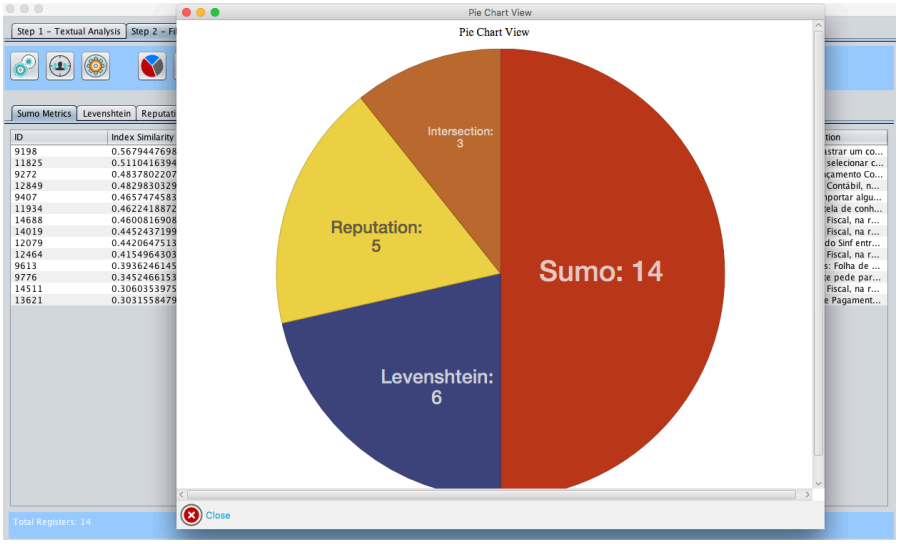


Figura 9: Visualização *Pie Chart* View

O *Pie Chart View* é um diagrama circular em que os valores de cada categoria estatística representada são proporcionais às respectivas medidas dos ângulos (1% no gráfico de setor equivale a 3,6º). Os dados percentuais serão distribuídos considerando-se a proporção da área a ser representada em relação aos valores das porcentagens. A área representativa no gráfico será demarcada conforme se pode observar na Figura 10.

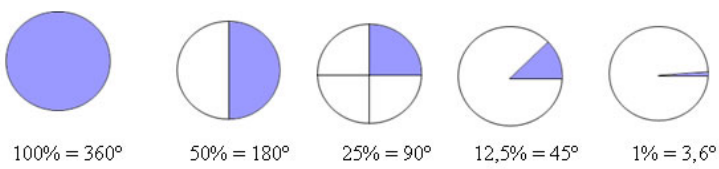


Figura 10: Área representativa do Gráfico

3.9 BUBBLE VIEW

A visualização *Bubble View*, apresentada na Figura 11, permite representar, de forma visual, o agrupamento das requisições de mudanças em cada uma das técnicas definidas no *framework*. O diagrama usa uma representação circular, tal como um cerco em volta de cada técnica, para que seja possível representar a hierarquia de valores de cada requisição. O tamanho do círculo e de cada nó revela uma dimensão quantitativa de cada ponto de dados em relação à hierarquia (técnica) selecionada. Os círculos envolventes mostram o tamanho cumulativo aproximado de cada sub-árvore de uma determinada técnica. Essa visualização é útil para definir quais requisições de mudança, dentro de cada técnica, são mais relevantes em questão de esforço gasto para sua execução. Dessa forma a distribuição dos valores dentro do gráfico, pode fornecer um auxílio na tomada de decisão ou na investigação de uma determinada requisição, visto que a mesma pode ser rejeitada pelo usuário.

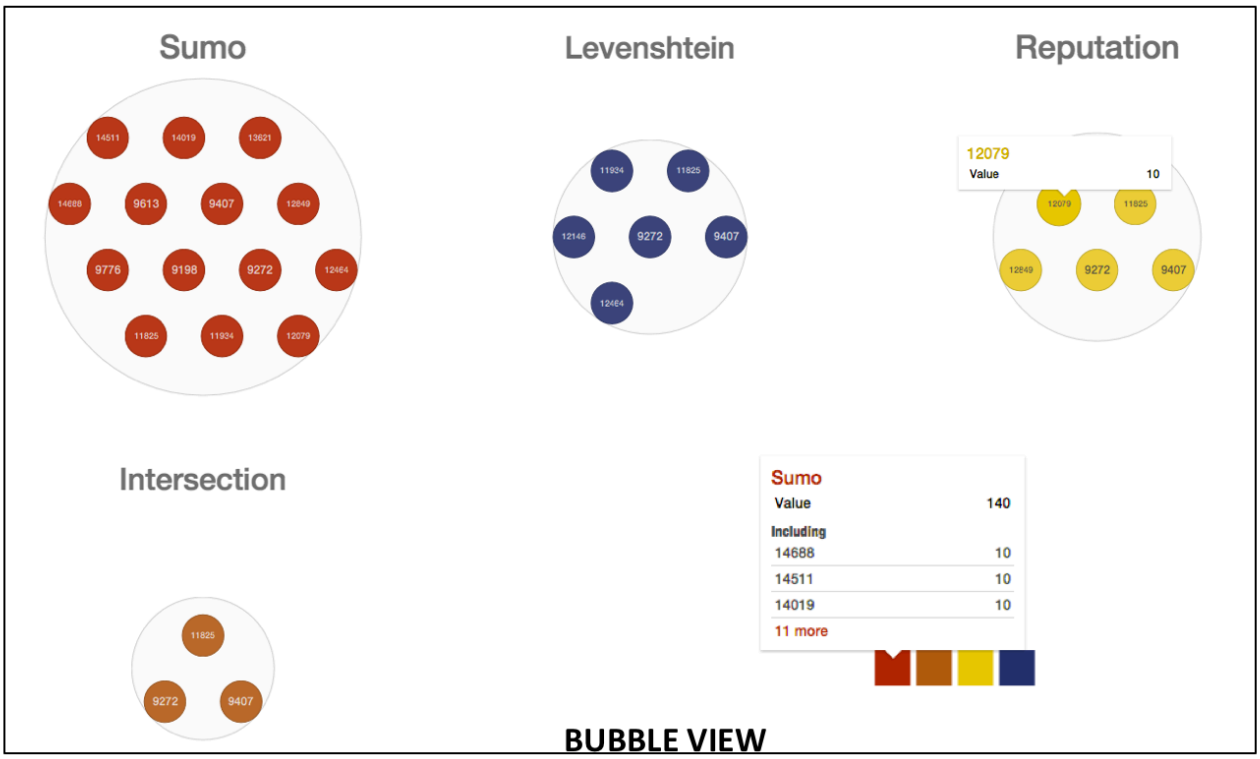


Figura 11: Visualização *Bubble View*

3.10 INTERACTIVE FORCE VIEW

A Interactive Force View, apresentada na Figura 12, permite representar, de forma visual, a nuvem de dados das requisições de mudanças, somando todas as técnicas definidas no framework. A parte mais importante desse gráfico é a ligação entre os nós ou requisições de mudanças, que fazem parte da interseção das técnicas. Essa ligação, ou a força interativa entre elas, permite estabelecer de forma visual, a relação entre as requisições de mudanças que fazem parte da interseção das técnicas e assim derivar observações por parte do usuário das mesmas. Essa visualização foi adicionada mediante a necessidade que foi apresentada à medida que se analisava a nuvens de dados. Buscou-se com isso identificar quais requisições possuíam ligações, permitindo assim selecionar apenas a fatia de dados que representa a ligação entre as requisições.

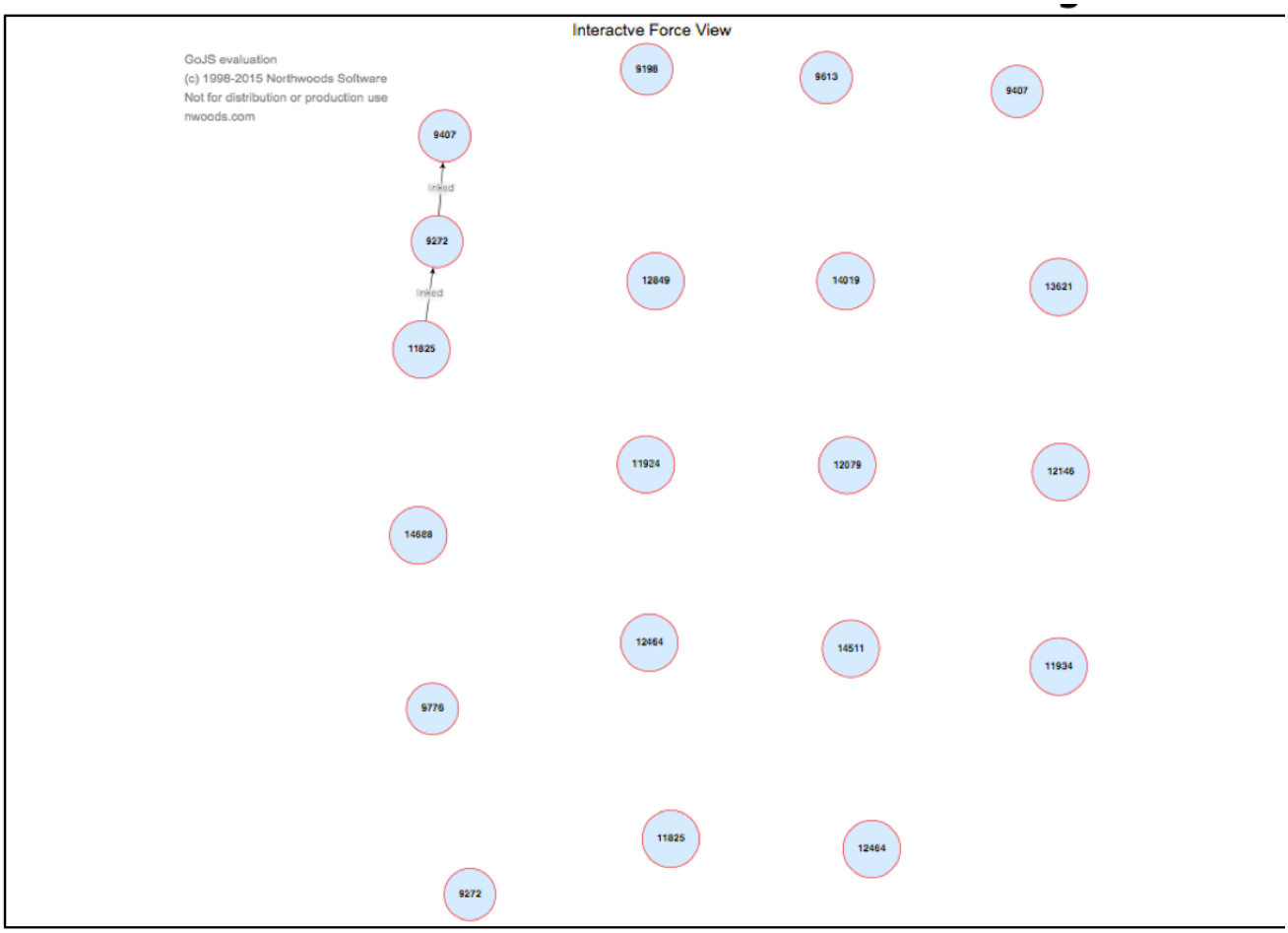


Figura 12: Visualização *Interactive Force View*

3.11 VEEN DIAGRAM VIEW

A representação visual do Veen Diagram View, disposta na Figura 13, permite ressaltar a relação de interseção entre as técnicas. O Diagrama de Veen (SHIN, 1994) consiste, basicamente, em círculos que possuem a propriedade de representar relações entre conjuntos numéricos. São utilizados para uma melhor visualização das propriedades dos conjuntos.

No contexto desde trabalho, o Veen Diagram View usa círculos sobrepostos para ilustrar as semelhanças, diferenças e relações entre as técnicas apresentadas pelo *framework*. As semelhanças entre as técnicas são representadas nas partes sobrepostas dos círculos, ou seja, a interseção das técnicas, enquanto as diferenças são representadas nas partes que não são sobrepostas. Pode-se visualizar em cada uma das partes do diagrama a quantidade de requisições de mudanças de cada técnica, bem como a quantidade de requisições que participam da interseção entre elas. Essa visualização foi adicionada à medida que se precisou de encontrar a interseção entre as técnicas e a quantidade de requisições de mudança pertencentes a interseção, e assim, de forma visual, perceber os totais apresentados na fatia de dados pertencentes à interseção e a quantidade total encontrada em cada uma das técnicas.

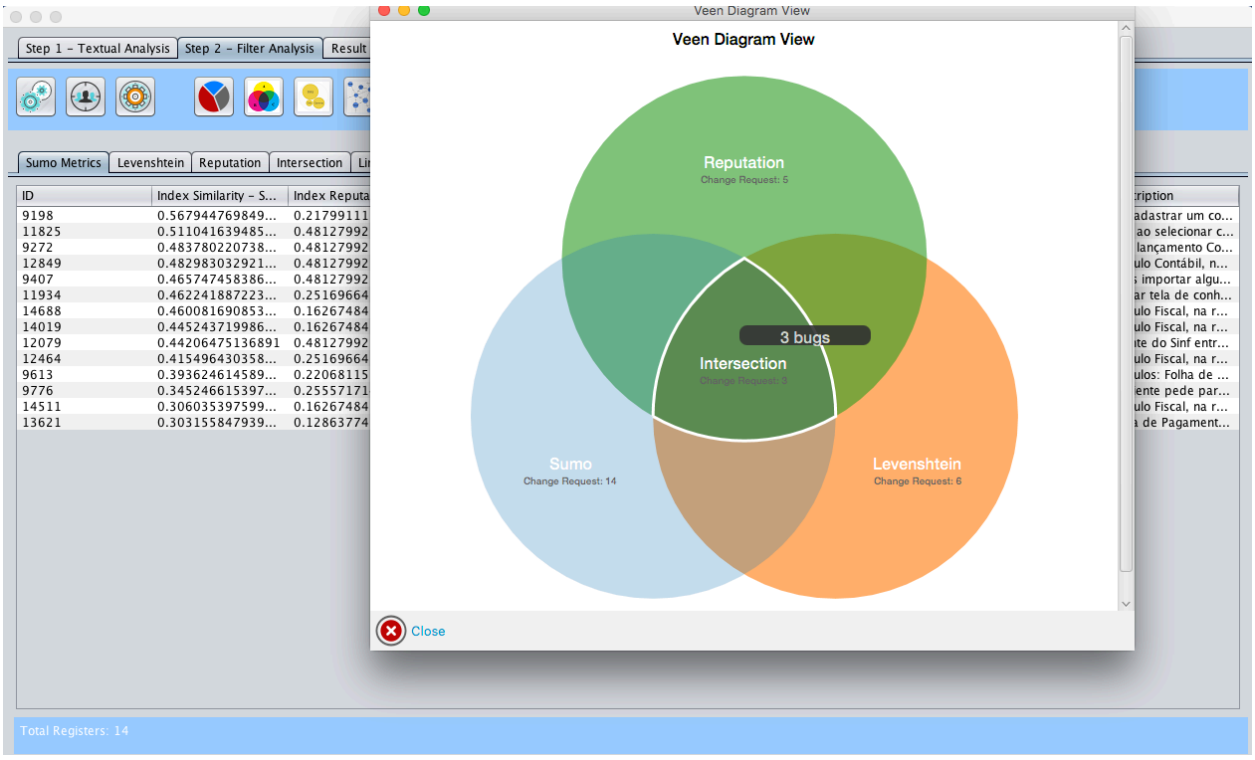


Figura 13: Visualização *Veen Diagram View*

3.12 BOX PLOT VIEW

Um *Box Plot* (MASSART, 2005) é um gráfico utilizado para avaliar a distribuição empírica dos dados. É formado pelo primeiro e terceiro quartil e pela mediana. As hastes inferiores e superiores se estendem, respectivamente, do quartil inferior até o menor valor, não inferior ao limite inferior, e do quartil superior até o maior valor, não superior ao limite superior. Os limites são calculados conforme abaixo:

$$Limite\ Inferior\colon\ \max\{\min(dados)\,;Q1-1{,}5(Q3-Q1)$$

$$Limite\ Superior\colon\ \min\{\max(dados)\,;Q3+\ 1{,}5(Q3-Q1)$$

Os pontos que estão fora destes limites são considerados *outliers* (MASSART, 2005), ou seja, dados muito diferentes do conjunto, capazes de levar o usuário a cogitar as suas eliminações ou uma melhor análise desses dados. Essa visualização é útil para definir quais requisições de mudança, dentro de cada técnica, são mais relevantes em questão de esforço gasto para sua execução. Dessa forma, a distribuição dos valores dentro do gráfico, pode fornecer um auxílio na tomada de decisão ou na investigação de uma determinada requisição, visto que a mesma pode ser rejeitada pelo usuário.

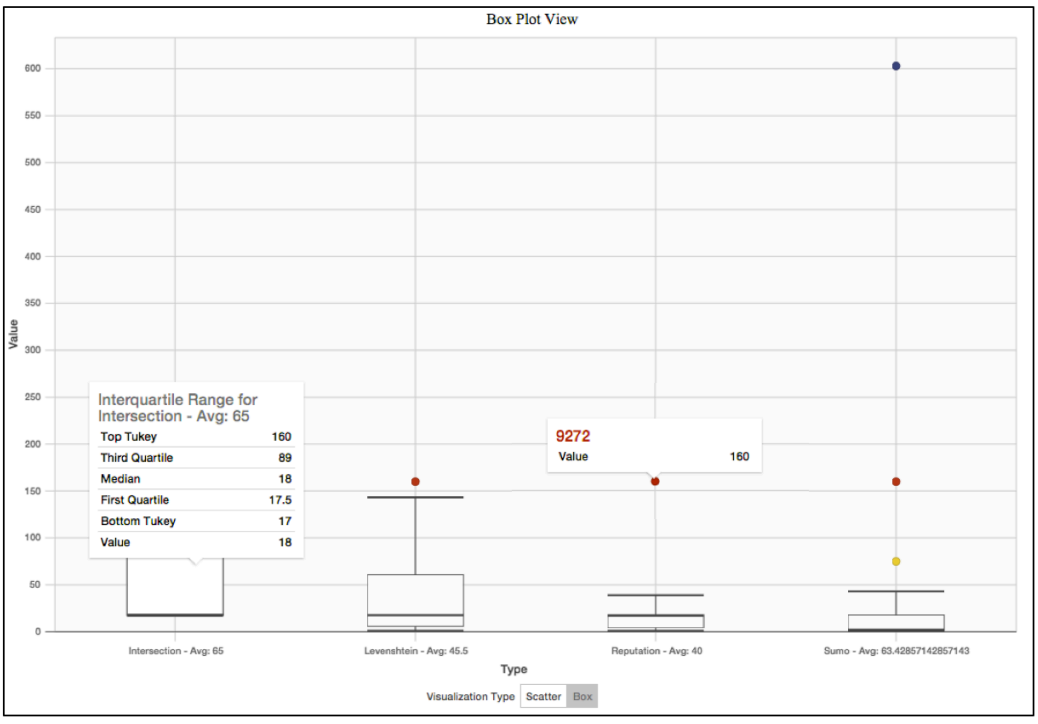


Figura 14: Visualização *Box Plot View*

Por meio dos cálculos obtidos pelo *Box Plot*, representados graficamente na Figura 14, é possível encontrar e remover *outliers* de forma automática, sendo que essa opção foi adicionada ao desenvolvimento do *framework*. A remoção de *outliers* pode significar dados mais homogêneos, pois pode-se remover todos os dados que são muito discrepantes do conjunto, capazes de alterar a precisão da estimativa de esforço.

3.13 REPUTATION DEVELOPERS VIEW

O gráfico *Reputation Developers View*, apresentado na Figura 15, permite representar, de forma visual, a reputação dos desenvolvedores em função do tempo. Essa informação pode ser útil para a tomada de decisão, à medida que torna possível analisar, de forma gráfica, como o índice da reputação de cada desenvolvedor está se comportando ao longo do tempo e como o mesmo se comporta em função dos demais desenvolvedores. Com essas informações o usuário do *framework* pode excluir ou permitir requisições de mudanças históricas de um determinado desenvolvedor ou aumentar o grau de confiança, para que apenas usuários melhores ranqueados apareçam na seleção de requisições de mudanças feitas pelo *framework*.

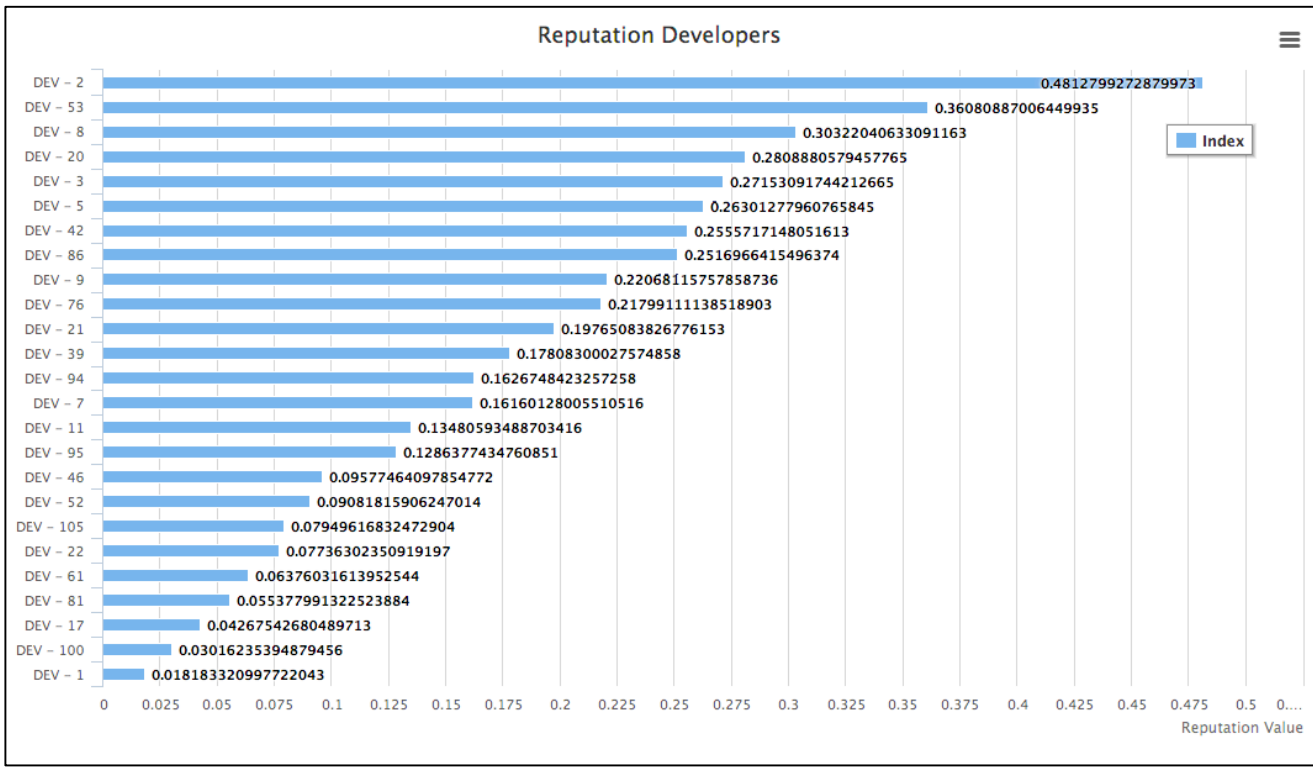


Figura 15: Visualização *Reputation Developers View*

3.14 LINEAR REGRESSION VIEW

A visualização *Linear Regression View*, demonstrada na Figura 16, permite a representação gráfica dos dados em um sistema cartesiano. No diagrama de dispersão dos dados, apresentado por essa visualização, é possível analisar o comportamento das requisições de mudanças em função do tempo. Esse gráfico ainda permite a análise da direção e de pontos discrepantes dos dados, que podem ser analisados individualmente e removidos, caso necessário, do cálculo da estimativa pelo usuário do *framework*. Quanto maior a correlação entre o esforço obtido nas requisições de mudança e seu tempo, mais próxima de uma reta a 45° *ou* 135° será a distribuição. Uma reta em 45° sugere que o esforço gasto nas requisições de mudanças está aumentando, já uma reta em 135° é um indicativo de que o esforço nas requisições de mudanças está diminuindo.

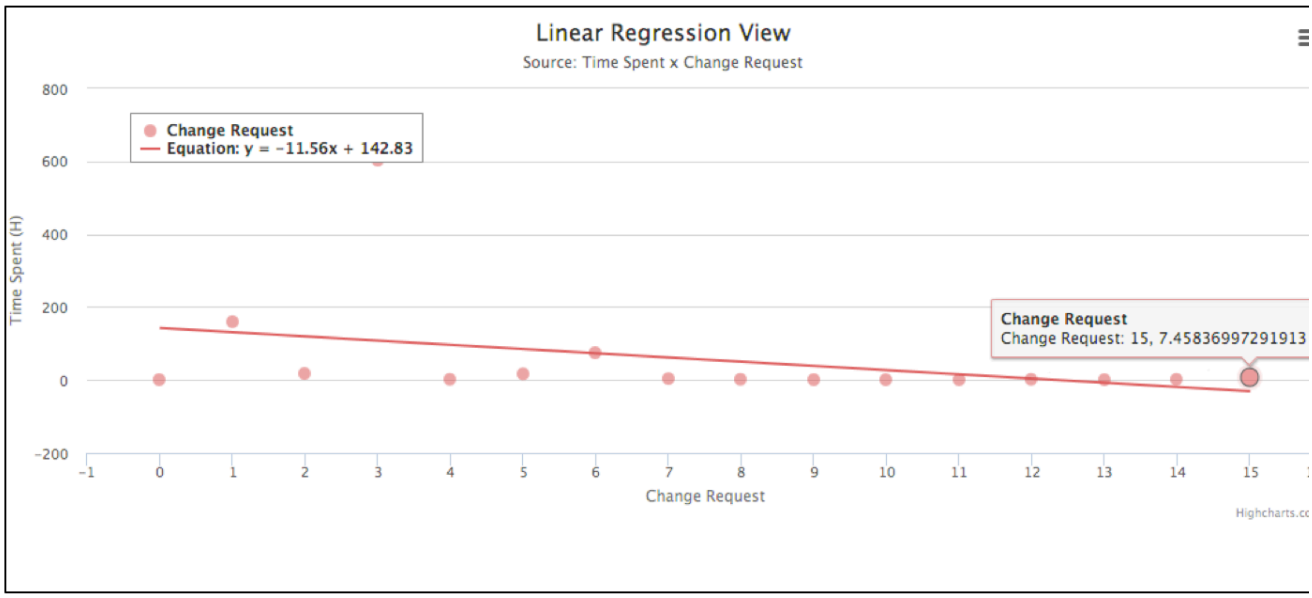


Figura 16: Visualização *Linear Regression View*

3.15 CODE ANALYSER VIEW

A visualização *Code Analyser View*, demonstrada na Figura 17, permite a representação gráfica das relações entre as requisições de mudança e as classes de código fonte envolvidas e alteradas por cada requisição histórica. As classes afetadas por cada requisição e suas interconexões, permitem ao usuário do *framework* analisar de forma gráfica o impacto da solicitação de mudança selecionada. Essa análise pode

auxiliar na tomada de decisão, por parte do usuário, do esforço calculado em funções das técnicas do *framework*. Essa visualização baseia-se na rastreabilidade de dados, levando em consideração o *log* das operações realizadas pela equipe de desenvolvimento junto ao servidor GIT da empresa.

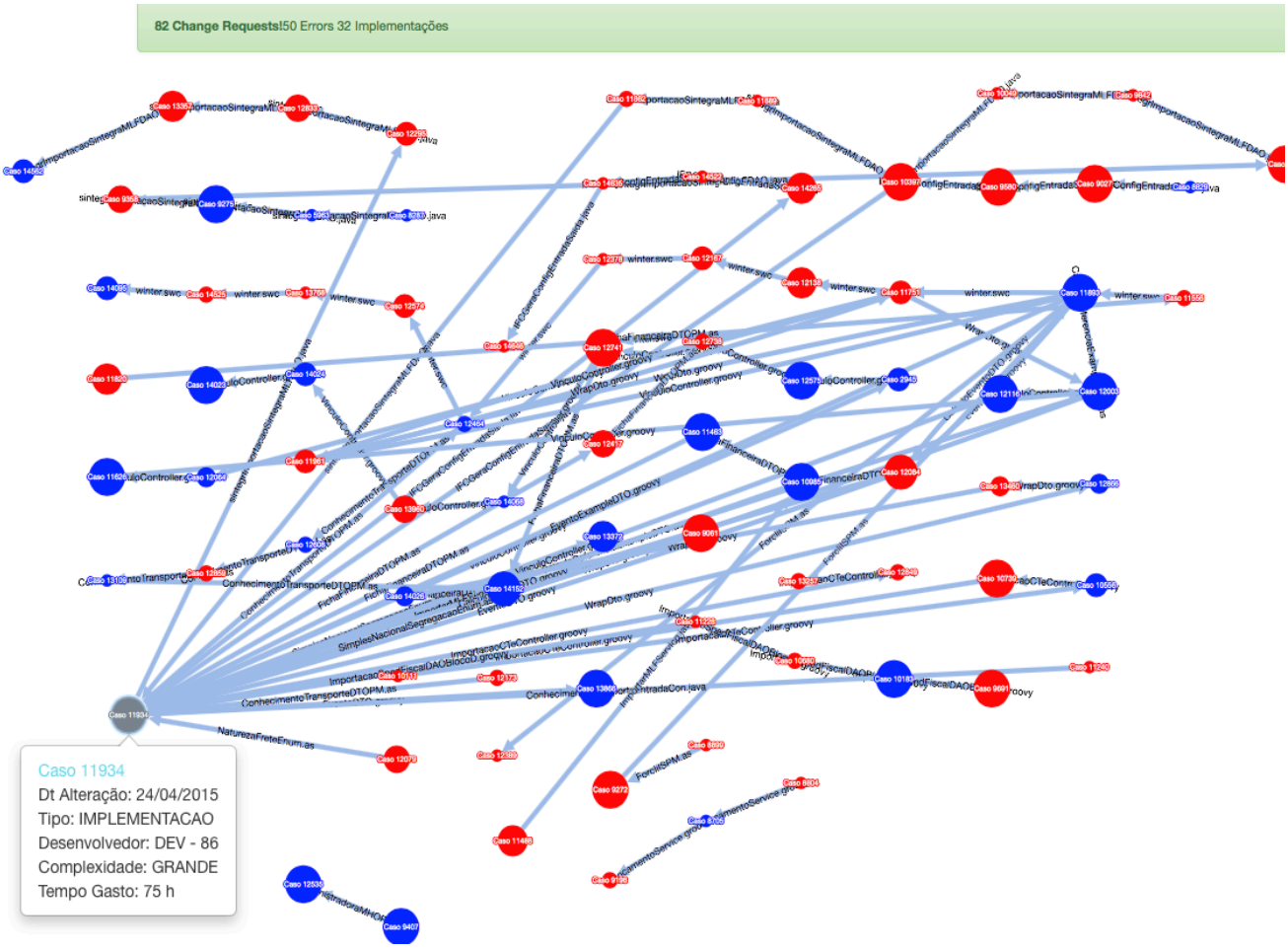


Figura 17: Visualização *Code Analyser View*

3.16 PLANNING

A última parte do *framework* é o componente *Planning,* disposto na Figura 18, o qual permite ao usuário (i) selecionar o cálculo de esforço efetuado para uma determinada requisição de mudança, utilizando as técnicas apresentadas na arquitetura, e (ii) armazenar esses dados para que seja possível, de forma incremental, gerar um *Sprint* ou manter um histórico dos cálculos efetuados para diversas requisições de mudanças.

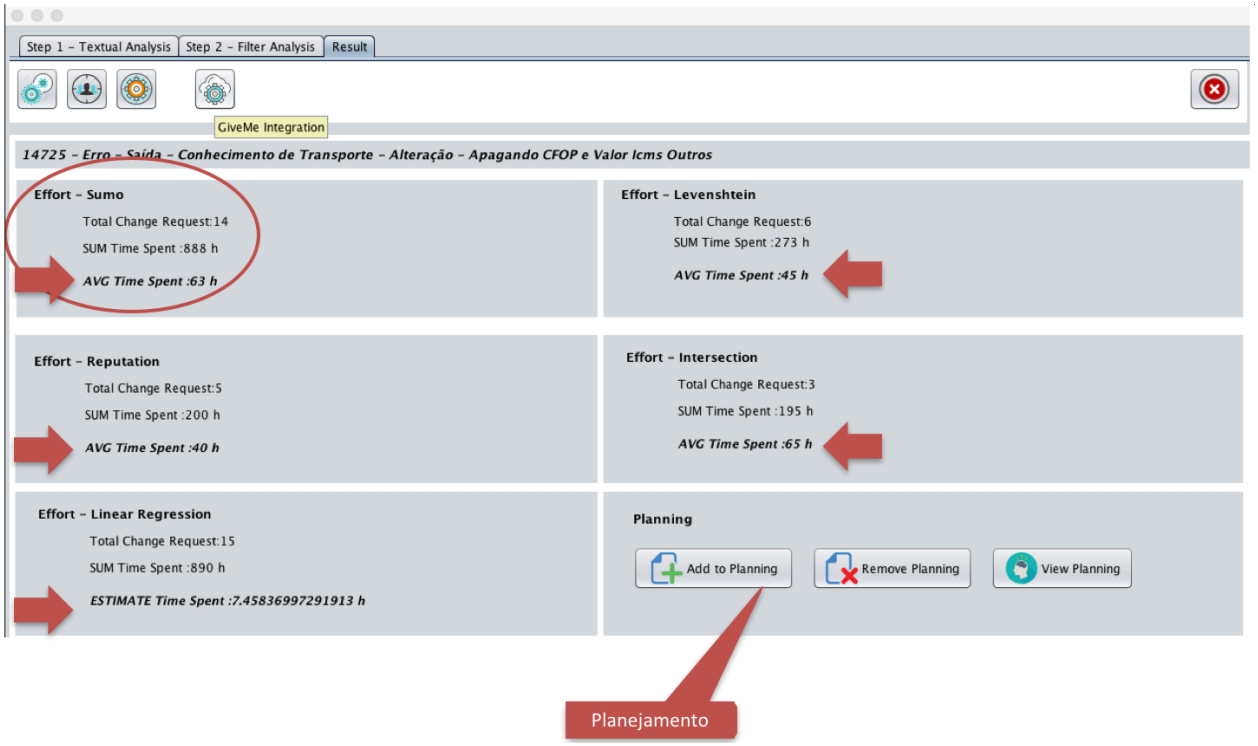
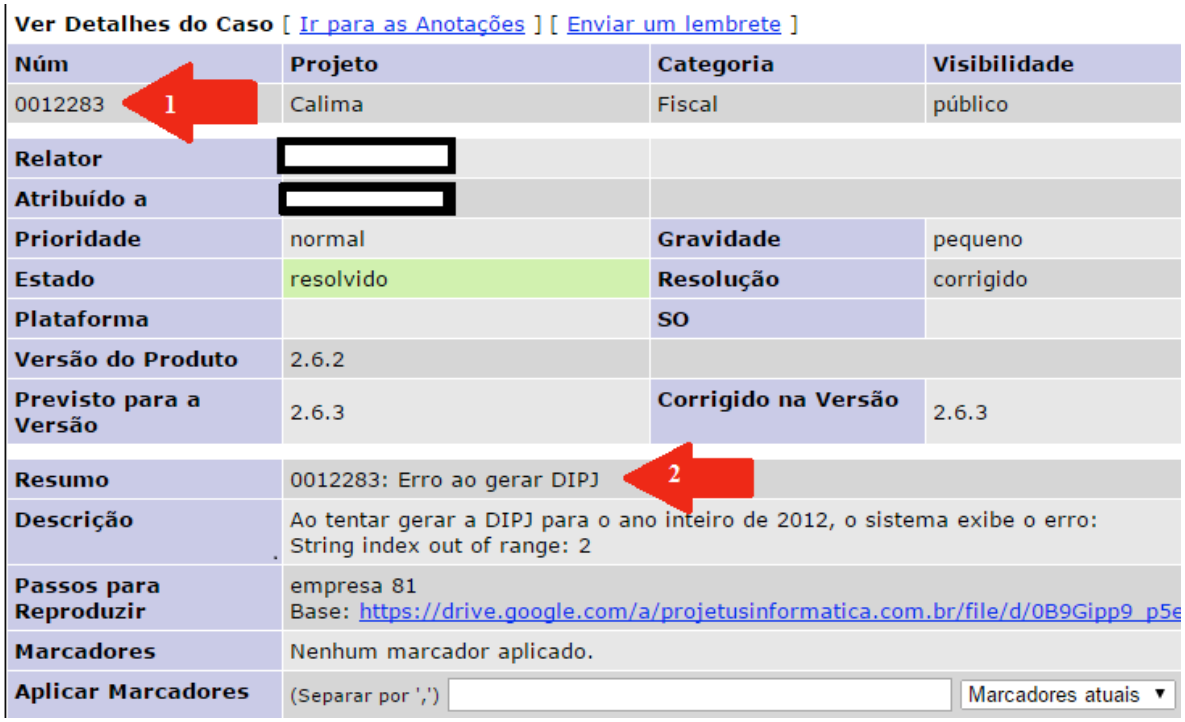


Figura 18: Componente *Planning*

A próxima subseção demonstrará um exemplo prático do funcionamento da arquitetura do *framework* e seus componentes.

3.17 EXEMPLO DE USO DA SOLUÇÃO PROPOSTA

Para demonstrar o uso do *framework*, em um contexto de desenvolvimento de software ágil, em que a equipe se reúne para definir a estimativa de esforço das requisições de mudanças solicitadas pelos clientes, foi elaborado um cenário, onde um usuário faz uma solicitação de mudança no software, conforme demonstrado na Figura 19. Em seguida, a equipe planeja as atividades a serem desenvolvidas. Para isso, é necessário que a ferramenta *GiveMe Infra* extraia do repositório de mudanças Mantis, todas as requisições ainda não implementadas. A requisição apresentada na Figura 18 relaciona-se à implementação de uma funcionalidade no software, que a equipe de desenvolvimento selecionou como parte do *Sprint*. O número da requisição (1) é armazenado junto com sua descrição (2).



Ver Detalhes do Caso [Ir para as Anotações] [Enviar um lembrete]

Núm	Projeto	Categoria	Visibilidade
0012283	Calima	Fiscal	público
Relator	[redacted]		
Atribuído a	[redacted]		
Prioridade	normal	**Gravidade**	pequeno
Estado	resolvido	**Resolução**	corrigido
Plataforma		**SO**	
Versão do Produto	2.6.2		
Previsto para a Versão	2.6.3	**Corrigido na Versão**	2.6.3
Resumo	0012283: Erro ao gerar DIPJ		
Descrição	Ao tentar gerar a DIPJ para o ano inteiro de 2012, o sistema exibe o erro: String index out of range: 2		
Passos para Reproduzir	empresa 81 Base: https://drive.google.com/a/projetusinformatica.com.br/file/d/0B9Gipp9_p5e		
Marcadores	Nenhum marcador aplicado.		
Aplicar Marcadores	(Separar por ',') Marcadores atuais		

Figura 19: Exemplo de Requisição de Mudança

O passo inicial, antes de se calcular o esforço para o caso selecionado, é configurar o *framework* para que o mesmo esteja de acordo com as preferências do usuário. A ferramenta permite a configuração de diversas opções, conforme demonstrado na Figura 20, para que a mesma seja adaptável às necessidades e opções de cada tipo de equipe e empresa. A opção *Find Type* (1), permite a escolha da análise de dados para aplicação das técnicas pelo sumário, descrição ou ambos os dados das requisições de mudança. A opção *Considered Years* (2) permite selecionar o tempo dos dados, em anos, a serem considerados na análise. A opção *Type Similarity Calculation* (3) permite escolher o tipo de técnica para o cálculo que será aplicado (Sumo, Levenshtein ou Reputation), bem como o grau de precisão que cada uma das técnicas terá. A opção *Linear Regression* (4) permite selecionar o cálculo de regressão linear, considerando-o para a estimativa de esforço. A opção *Remove All Outliers* (5) permite que todos os *outliers* existentes em cada técnica aplicada, sejam removidos automaticamente do cálculo do esforço, ou seja, todo e qualquer *outliers* é ignorado para o cálculo e desprezado na visualização dos dados.

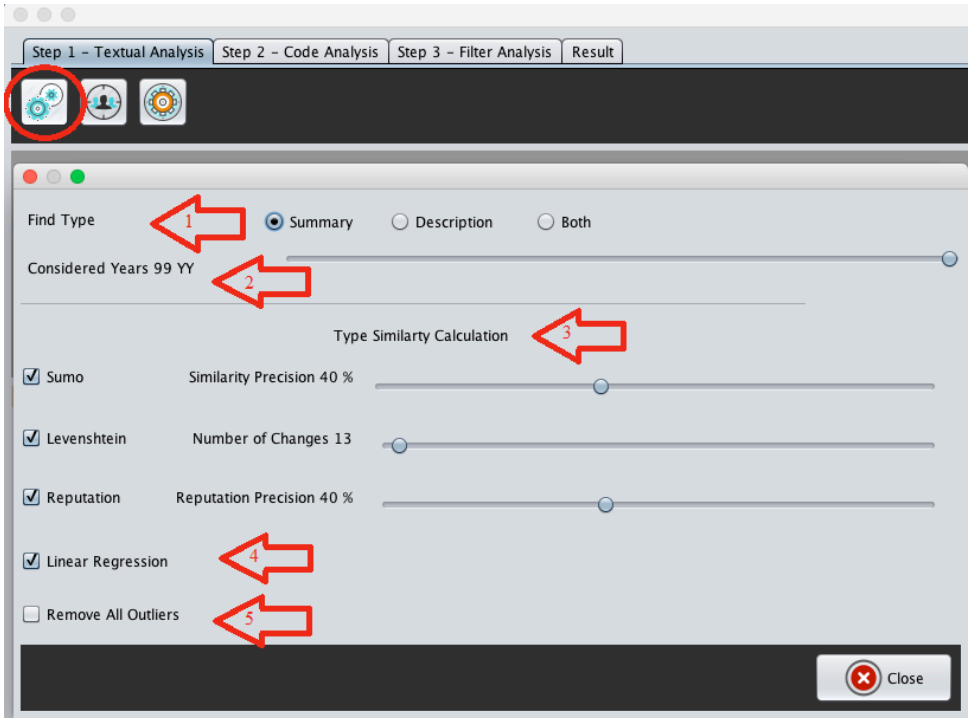


Figura 20: *GiveMe Effort* - Opções de configuração e calibração do *framework*

A tela inicial do *framework GiveMe Effort*, apresentado na Figura 21, permite a seleção da requisição de mudança (1) ao clicar no botão (2), iniciando-se uma busca nos repositórios de dados pelas requisições que textualmente se assemelham a requisição selecionada para análise. Em seguida, é apresentado, na guia (3) de cada uma das técnicas analisadas (*Sumo, Levenshtein, Reputation* e *Linear Regression*), o total de registros (4) encontrados e todos os detalhes a respeito das requisições selecionadas.

É possível detalhar cada requisição, conforme demonstrado na Figura 22, para uma melhor análise dos dados, clicando-se duas vezes em cima de cada requisição. A janela apresenta o sumário (1) e a descrição completa (2) da requisição de mudança, bem como, o desenvolvedor (3) e o tempo (4) gasto, ou seja, o esforço despendido para o desenvolvimento da requisição de mudança do software em análise.

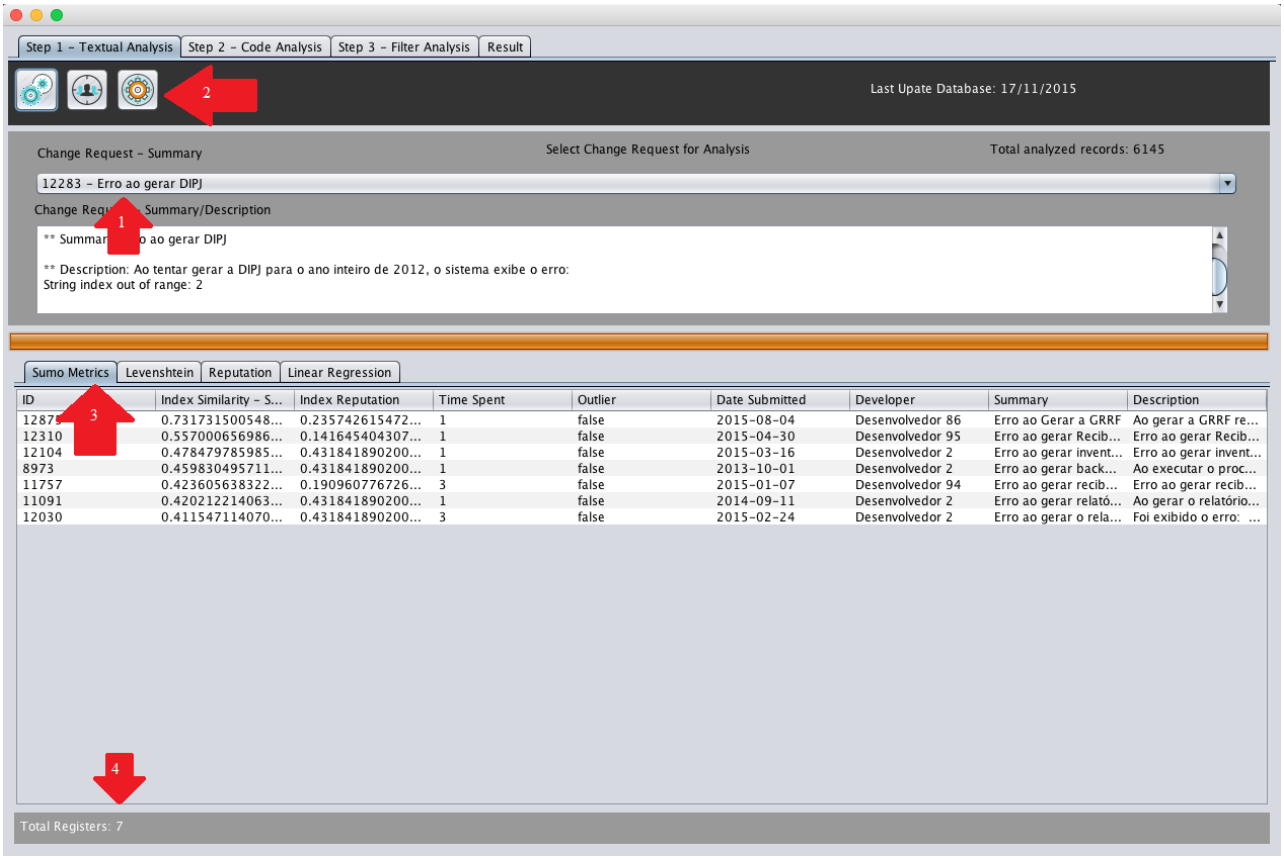


Figura 21: *GiveMe Effort* - Passo 1 – Análise Textual

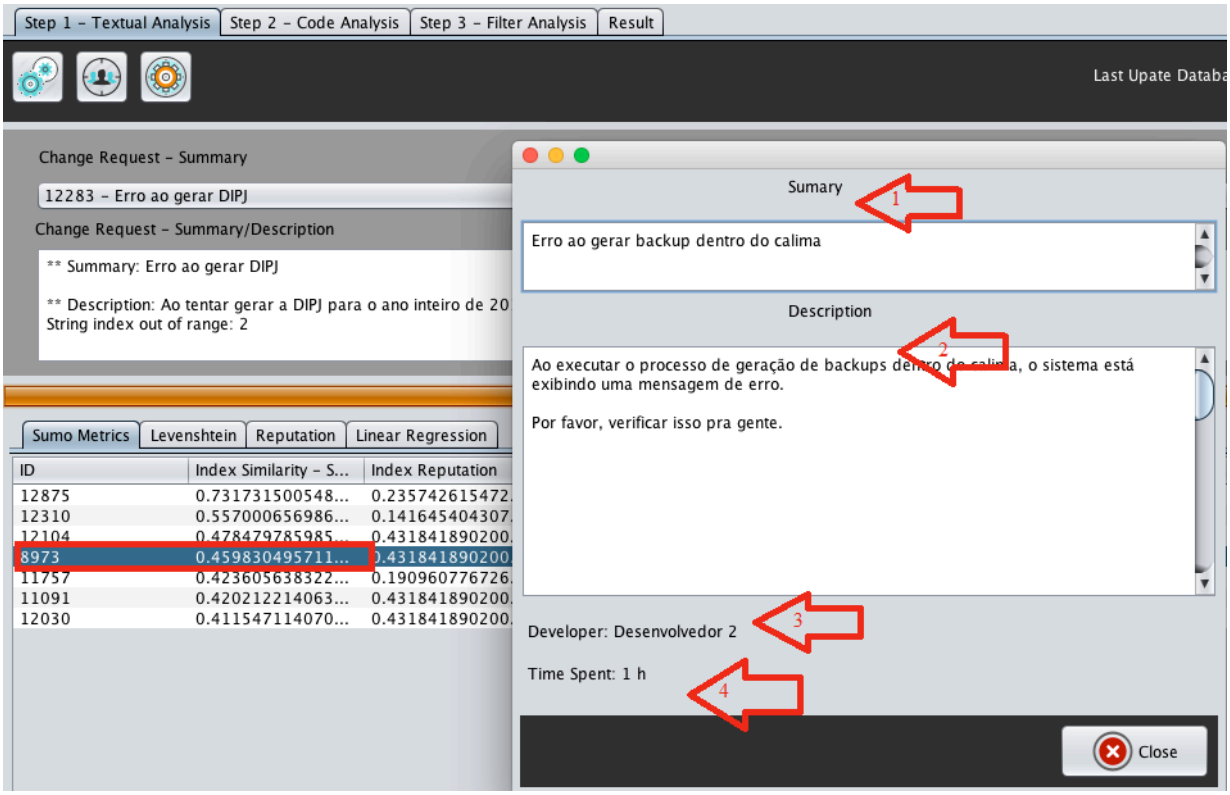
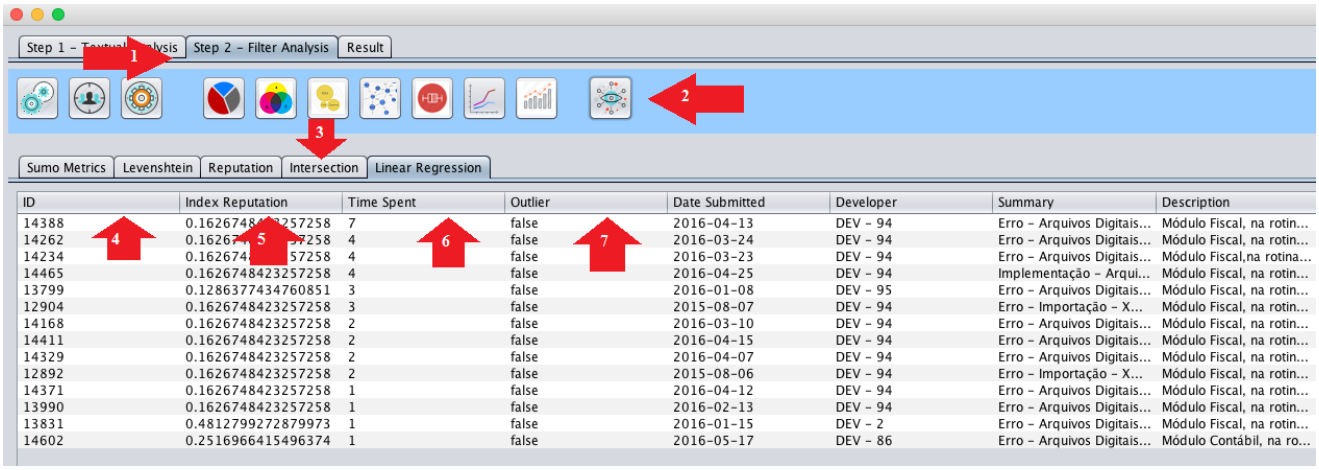


Figura 22: Análise dos dados da requisição de mudança

O *framework* permite filtrar requisições de mudança processadas, por meio da guia *Filter Analysis* (Fig. 23) (1), possibilitando a exclusão de uma requisição, caso o usuário julgue necessário. Esta guia disponibiliza várias visualizações (2), que serão apresentadas a seguir, dada a análise solicitada para a requisição de mudança, bem como a interseção entre as técnicas (3). Pode-se ainda considerar o número da requisição que foi selecionada para análise de esforço (4), o índice de similaridade da mesma (5), o índice de reputação aplicado ao desenvolvedor (6) e o tempo gasto (7) para o desenvolvimento da atividade, bem como se essa requisição é um *outliers* (8) em relação às demais selecionadas.



ID	Index Reputation	Time Spent	Outlier	Date Submitted	Developer	Summary	Description
14388	0.16267484[illegible]257258	7	false	2016-04-13	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
14262	0.16267[illegible]257258	4	false	2016-03-24	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
14234	0.16267[illegible]257258	4	false	2016-03-23	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal,na rotina...
14465	0.1626748423257258	4	false	2016-04-25	DEV - 94	Implementação - Arqui...	Módulo Fiscal, na rotin...
13799	0.1286377434760851	3	false	2016-01-08	DEV - 95	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
12904	0.1626748423257258	3	false	2015-08-07	DEV - 94	Erro - Importação - X...	Módulo Fiscal, na rotin...
14168	0.1626748423257258	2	false	2016-03-10	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
14411	0.1626748423257258	2	false	2016-04-15	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
14329	0.1626748423257258	2	false	2016-04-07	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
12892	0.1626748423257258	2	false	2015-08-06	DEV - 94	Erro - Importação - X...	Módulo Fiscal, na rotin...
14371	0.1626748423257258	1	false	2016-04-12	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
13990	0.1626748423257258	1	false	2016-02-13	DEV - 94	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
13831	0.4812799272879973	1	false	2016-01-15	DEV - 2	Erro - Arquivos Digitais...	Módulo Fiscal, na rotin...
14602	0.2516966415496374	1	false	2016-05-17	DEV - 86	Erro - Arquivos Digitais...	Módulo Contábil, na ro...

Figura 23: Guia *Filter Analysis*

Para a requisição selecionada, e com a configuração selecionada na Figura 18, as seguintes análises gráficas foram obtidas:

- O *Pie Chart*, apresentado na Figura 24, demonstrou que quatro requisições foram encontradas na técnica de Reputação, oito requisições de mudanças com a técnica Levenshtein, sete requisições com a técnica Sumo e uma requisição que participou da interseção de todas as demais técnicas. Pode-se perceber que a técnica, nesse caso específico, que mais possui requisições de mudança é a Levenshtein, o que pode ser um indicativo de um ajuste nas configurações para que a quantidade de trocas esperadas na técnica seja menor, ou seja, uma maior proximidade entre o texto da requisição de mudança selecionada e as demais encontradas.

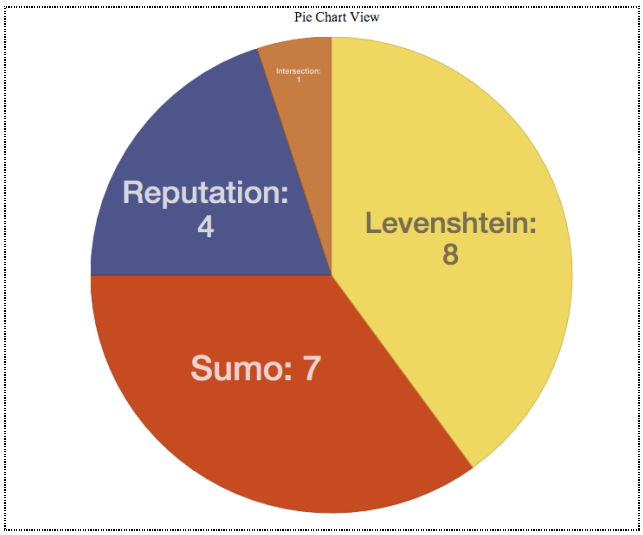


Figura 24: Visualização *Pie Chart* gerada pelo exemplo de funcionamento da proposta

- Uma análise dos dados na visualização de regressão linear, apresentada na Figura 25, mostra que na massa de dados selecionada há uma tendência dos valores de se aproximarem de zero, ou seja, é um indicativo de que o esforço nas requisições de mudanças está diminuindo.

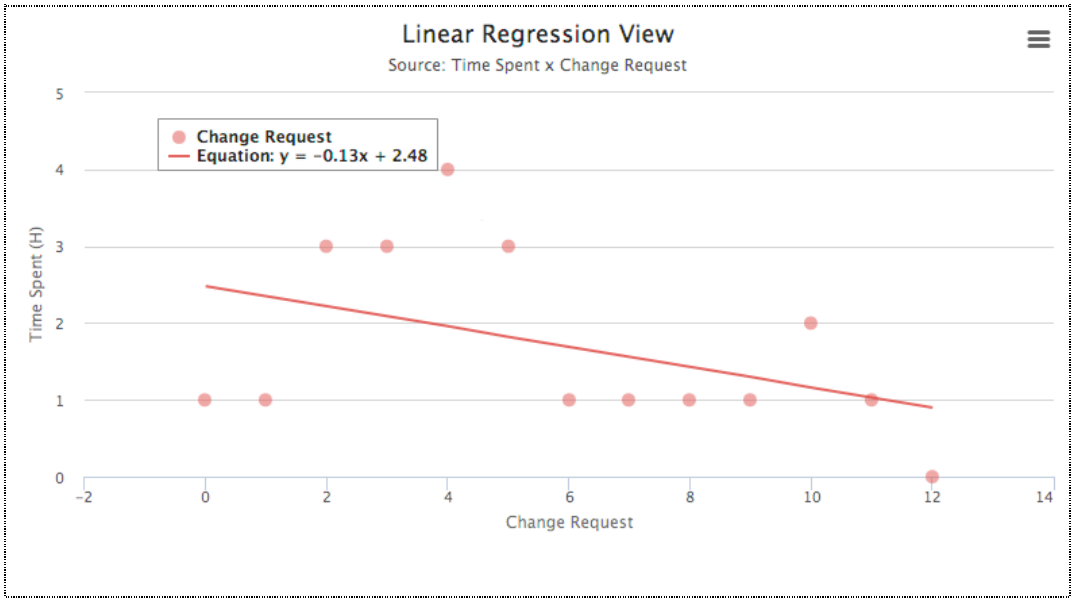


Figura 25: Visualização *Linear Regression* gerada pelo exemplo de funcionamento da proposta

- Uma análise dos dados na visualização *Bubble View* (Figura 26), demonstra as requisições de mudança com maior peso (tempo gasto) em

cada técnica apresentada. Pode-se notar que na técnica *Reputation* a requisição 12030 tem maior peso (3 h) que as demais (1 h). O mesmo se dá com a técnica Levenshtein, as requisições 11886 (4 h) e 11980 (3 h) tem maior peso que as demais requisições encontradas na análise de similaridade. Analisando a técnica Sumo, pode-se observar que as requisições 12030 e 11757 possuem o peso igual a 3 h em relação às demais de peso 1 h. Com isso, pode-se visualizar quais requisições estão afetando mais o cálculo do esforço final em cada técnica.

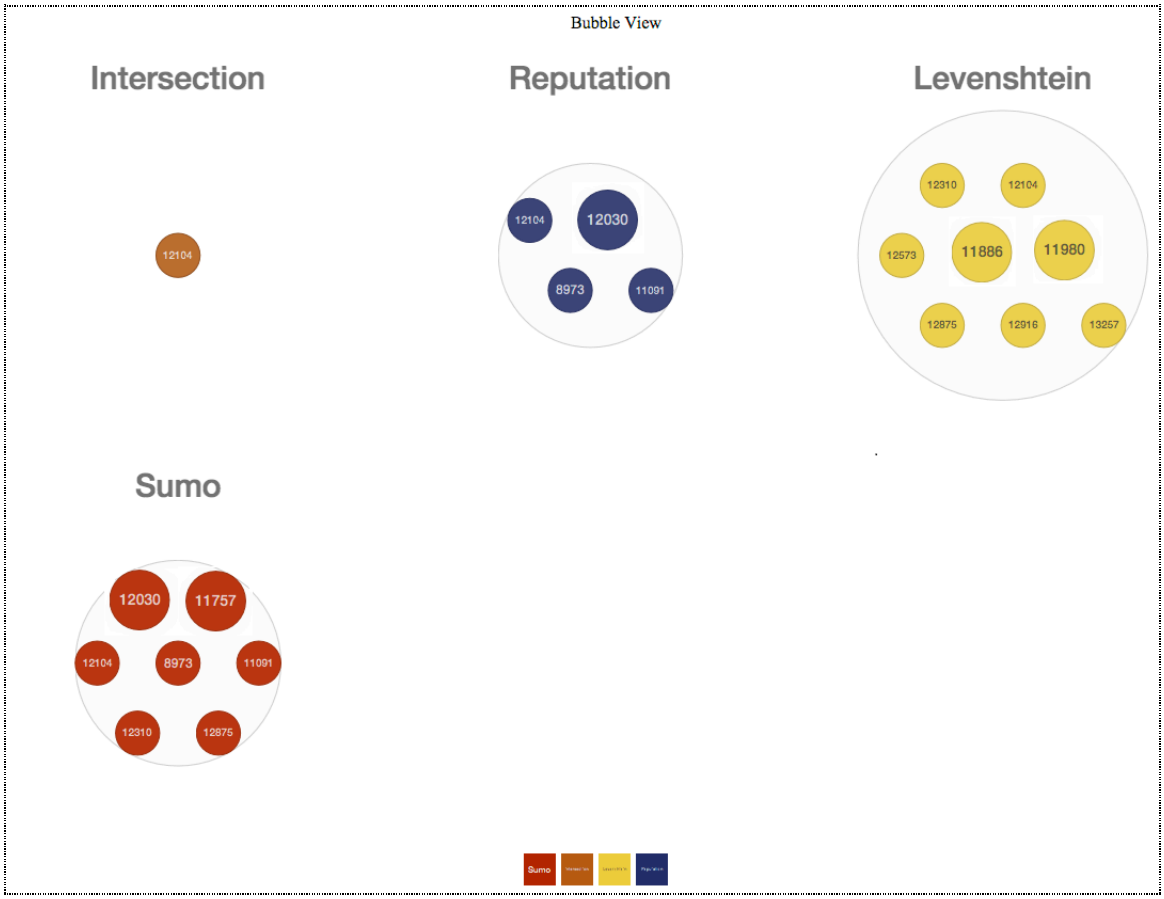


Figura 26: Visualização *Bubble View* gerada pelo exemplo de funcionamento da proposta

- A reputação dos desenvolvedores, demonstrada na Figura 27, apresentou que os cinco desenvolvedores com maior confiança na equipe de desenvolvimento são os desenvolvedores 2, 1, 86, 94 e 95. Analisando as requisições de mudanças selecionadas pelas técnicas, pode-se notar que o desenvolvedor 2 está presente em pelo menos 1 mudança em cada

técnica, indicando que é possível ter um grau de confiança maior em relação ao tempo gasto na conclusão das atividades, por tratar-se de um desenvolvedor mais experiente e com maior reputação na equipe. Essa análise auxilia no processo de confiança da estimativa final, visto que, os melhores desenvolvedores foram selecionados nas análises feitas pelo *framework.*

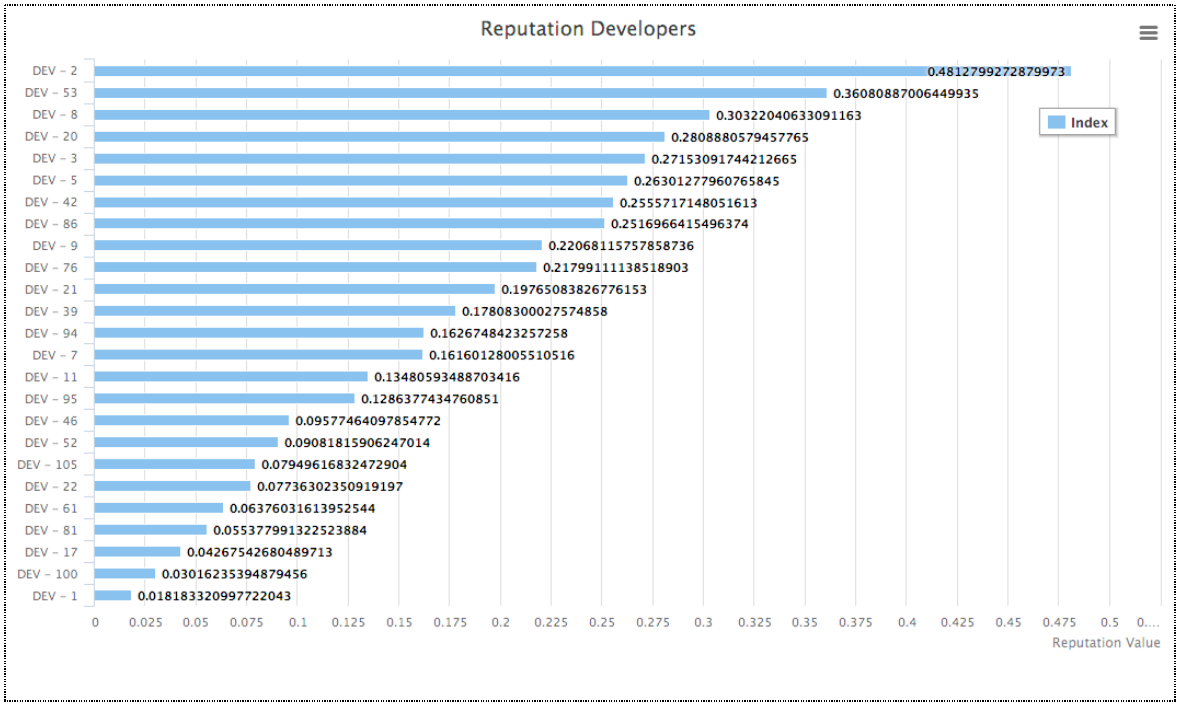


Figura 27: Visualização *Reputation Developers View* gerada pelo exemplo de funcionamento da proposta

- O Diagrama de *Veen,* exibido na Figura 28, apresenta as técnicas obtidas pela análise histórica de requisições. Nele pode-se observar a quantidade de requisições de mudança de cada técnica, bem como a quantidade de requisições em interseção.

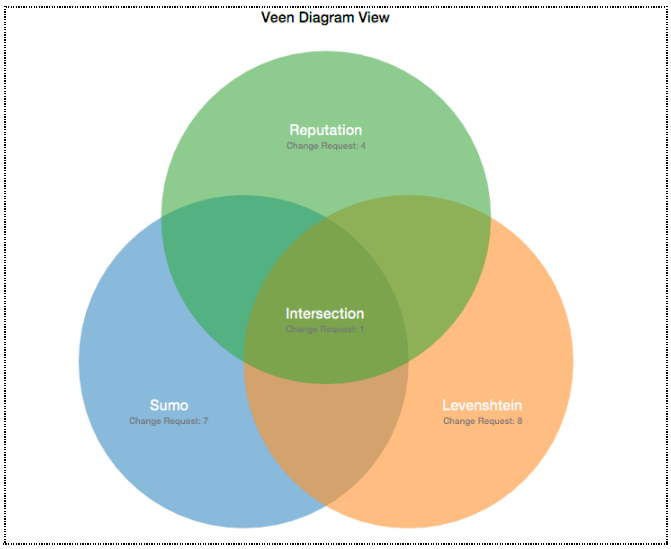


Figura 28: Visualização *Veen Diagrama View* gerada pelo exemplo de funcionamento da proposta

- Como o número de requisições de mudança é muito pequeno o *Box Plot View* não foi apresentado por não ter relevância do ponto de vista estatístico. A visualização *Interactive Force* não apresentou nenhum valor significativo, visto que o número de interseções, para esse caso específico, é igual a 1(um).
- A visualização Code Analyser, exibida na Figura 29, apresenta uma análise das relações das requisições de mudança bem como, as classes do código fonte afetados por cada requisição. Permite ao usuário ter uma visão do impacto da mudança em outras partes do sistema, bem como, que outras requisições estão relacionadas a referida solicitação de mudança.

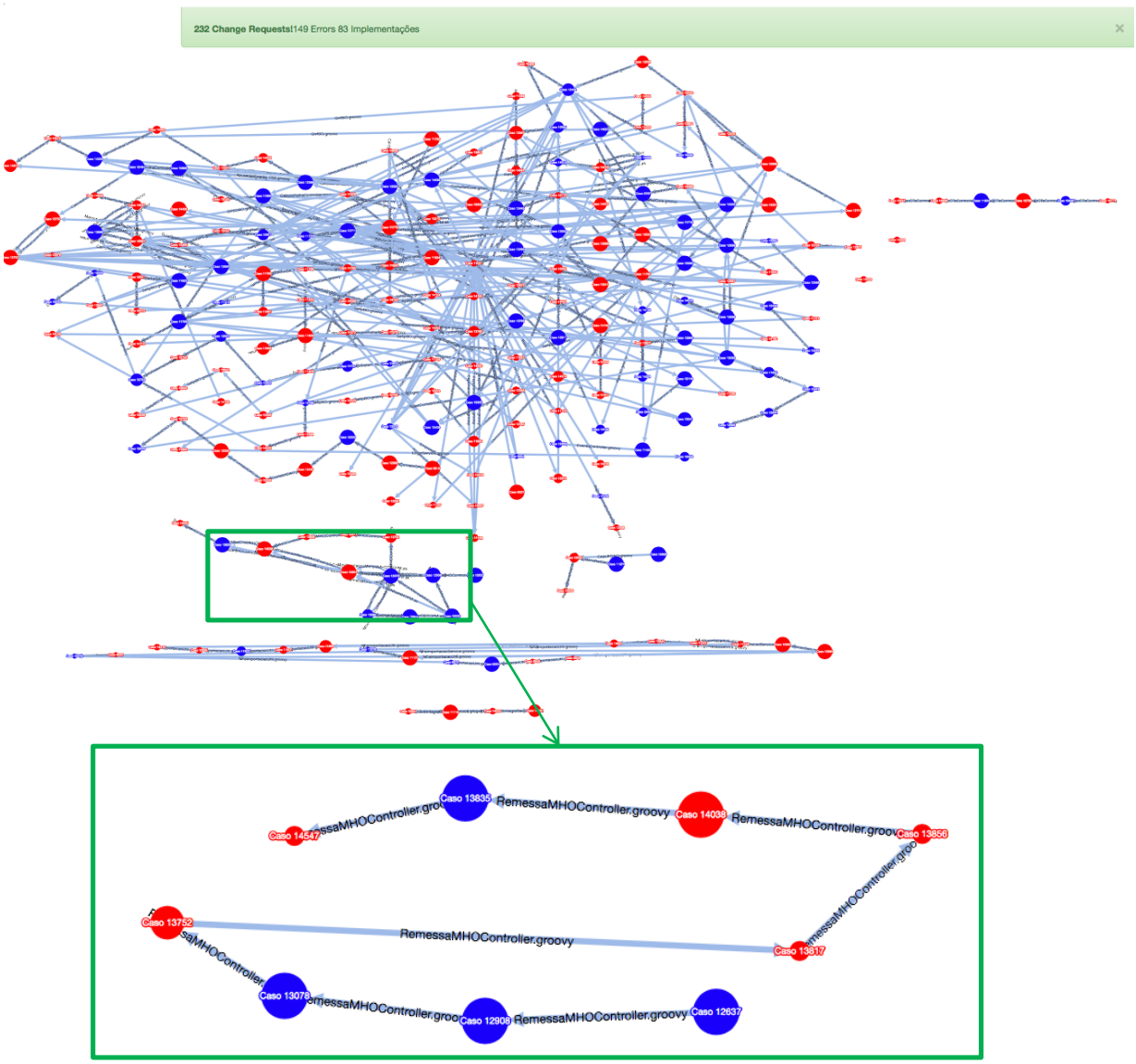


Figura 29: Visualização *Code Analyser* gerada pelo exemplo de funcionamento da proposta

Após as análises gráficas e as devidas exclusões de requisições, caso seja necessário, pode-se visualizar o resultado final das estimativas calculadas na guia *Result*, apresentada na Figura 30. O painel *Effort – Sumo* (1), apresenta o total de requisições selecionadas, a soma geral do tempo e a média do tempo gasto com a estimativa final para essa técnica. O painel *Effort – Reputation* (2) demonstra o cálculo do esforço total e a media das requisições de acordo com a reputação dos desenvolvedores. O painel *Effort – Levenshtein* (3) apresenta os mesmos dados das duas técnicas anteriores e a média do tempo gasto com a estimativa final. Igualmente, o painel *Effort – Intersection* (4) apresenta os mesmos dados e valores calculados na

interseção das técnicas, bem como sua média final com estimativa. Já o painel *Effort – Linear Regression* (5) demonstra o total de requisições encontradas por essas técnicas, o somatório geral do tempo encontrado e a estimativa, utilizando a regressão linear. O último painel *Planning (6),* apresenta as opções de adição da estimativa atual ao planejamento de um *Sprint* e guarda essas informações, de forma histórica, na base interna do *framework*. A opção (7) permite a integração de dados entre o *framework GiveMe Effort* e a infraestrutura do *GiveMe View*, conforme apresentado pela arquitetura proposta. Os dados coletados podem ser utilizados pela infraestrutura para gerar informações auxiliares em suas análises de código fonte e visualizações permitidas aos usuários da mesma.

Pode-se ainda remover o planejamento, caso o mesmo já tenha sido adicionado, ou visualizar todas as estimativas já realizadas e adicionadas ao planejamento.

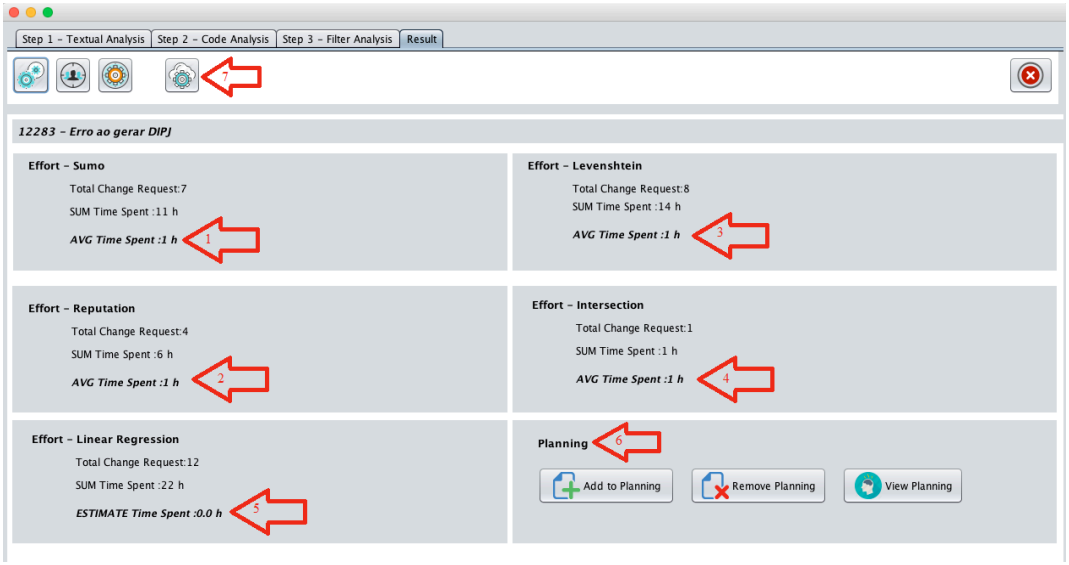


Figura 30: Guia *Result* do *framework* gerada pelo exemplo de funcionamento da proposta

A estimativa de esforço calculada pelo *framework*, não ultrapassou 1h de desenvolvimento nas técnicas apresentadas (Sumo, Levenshtein, Reputation, Intersection e Linear Regression). Como os dados que estão sendo analisados são de uma base real, de um software em desenvolvimento, foi possível acessar o valor realmente gasto pelo desenvolvedor e sua estimativa, de forma manual, para o caso apresentado na Figura 31. Pode-se notar que o tempo total gasto no desenvolvimento dessa requisição de mudança foi de 1h e 20min e o tempo estimado pelo desenvolver,

de forma *ad-hoc*, foi de 3h. O cálculo da estimativa (1h) feita pelo *framework* se aproximou bastante do valor real gasto (1h 20min), demonstrando a eficiência de uma análise histórica dos dados com as técnicas apresentadas.

Figura 31: Estimativa x Tempo Gasto real da solicitação analisada

A Figura 32, apresenta o fluxo do funcionamento do *framework* em três passos que compreendem da análise textual a entrega dos resultados. O primeiro passo (*Textual Analyser*) permite selecionar as requisições que serão analisadas para a mudança selecionada pelo usuário para análise. O segundo passo (*Filter Analysis*) permite o filtro e acesso às visualizações propostas pela solução e, finalmente, o último passo (*Result*), permite analisar os resultados encontrados e a adição das estimativas ao planejamento da equipe de desenvolvimento. Não existe uma interação entre a calibração do *framework* e as visões. Como resultado, não permite que um *feedback* seja fornecido à medida que se altera uma calibração, tornando necessário o processo de busca textual a cada nova calibração.

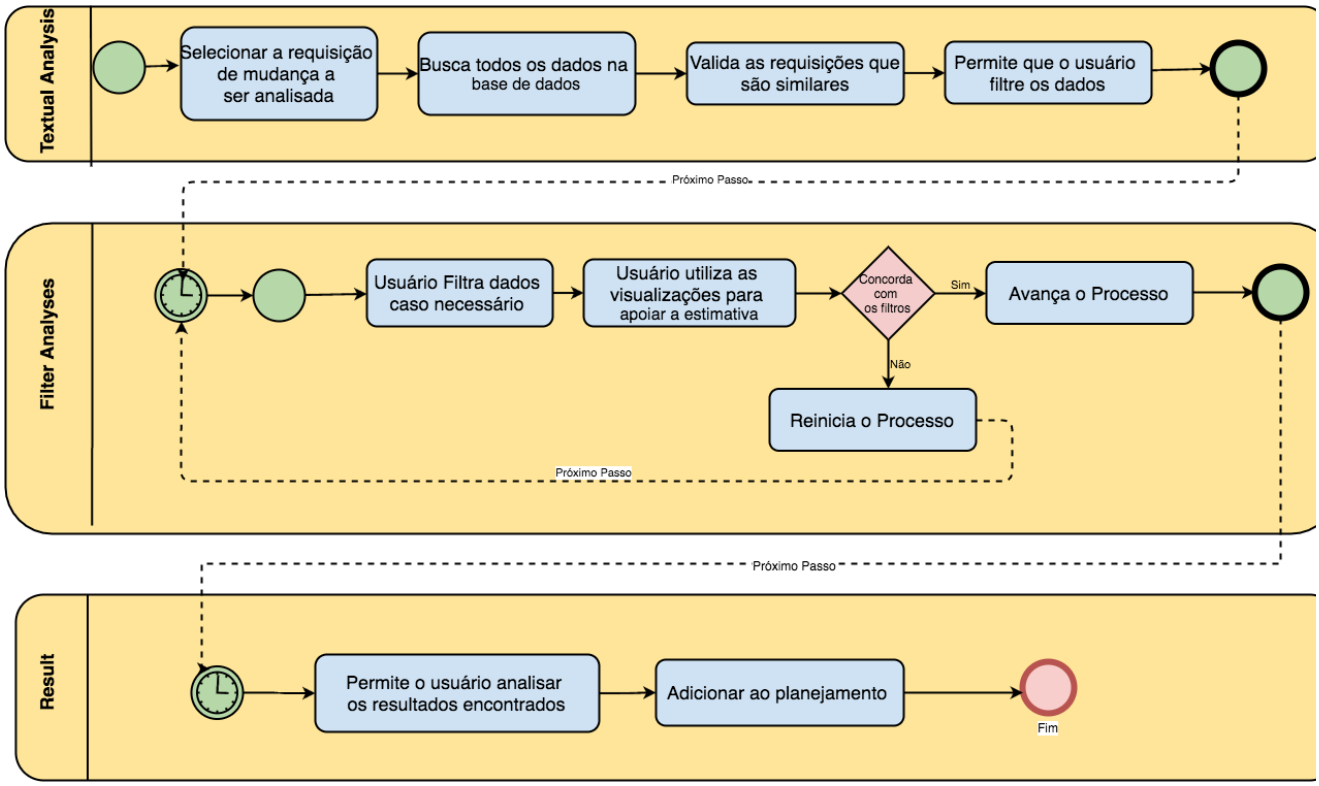


Figura 32: Fluxo de funcionamento geral da arquitetura do *GiveMe Effort*

3.18 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Dados os problemas levantados e a proposta do *framework* apresentada, foi detalhada nesse capítulo a arquitetura do *framework*, bem como demonstrado um exemplo de funcionamento da mesma em uma massa de dados real, de um software em desenvolvimento. A próxima seção apresentará o funcionamento do *framework* em um contexto real da empresa de desenvolvimento parceira, bem com os resultados obtidos nessa análise.

O próximo capítulo apresentará o *framework GiveMe Effort*, por meio de um exemplo prático de funcionamento, com dados reais fornecidos por uma empresa parceira.

4. GIVEME EFFORT – FUNCIONAMENTO DO FRAMEWORK E MEDIÇÕES

A presente seção tem por objetivo apresentar o funcionamento do *framework* e o resultado de medições realizadas na mesma, em função de *Sprints* realizados pela equipe de desenvolvimento.

A fim de verificar a viabilidade de uso da solução, bem como a sua aplicabilidade, foi utilizada uma base de dados (fontes, requisições de mudanças), dos últimos 5 (cinco) anos, da empresa parceira. Com isso, foi analisar o repositório de dados históricos, e assim verificar a viabilidade de uso da solução em um contexto real de utilização. A empresa parceira utiliza atualmente a metodologia de desenvolvimento SCRUM (SCHWABER, 2015), que se trata de um framework de desenvolvimento iterativo e incremental, utilizado no gerenciamento de projetos e desenvolvimento de software ágil. Nesse contexto, foi proposto e realizado várias medições para analisar o comportamento e funcionamento do *framework*, tendo como base um *Sprint* definido e estimado pela equipe de desenvolvimento atual da empresa parceira.

4.1 FUNCIONAMENTO DO FRAMEWORK

O primeiro passo para demonstrar o funcionamento do *framework* foi a definição de um *Sprint*, para o qual a equipe de desenvolvimento da empresa parceira, precisasse planejar a estimativa de esforço. Em seguida, foi realizado o respectivo cálculo utilizando o *framework GiveMe Effort*. A equipe de desenvolvimento em sua reunião de planejamento definiu, de forma manual, a estimativa de esforço de cada requisição de mudança, embasando-se na técnica *Planning Poker* (SCHWABER, 2015), conforme o procedimento padrão da empresa. Após essa coleta de informações a equipe deu início ao desenvolvimento do *Sprint* e, após a sua conclusão, foi mensurado o tempo real gasto para cada requisição de mudança. O resultado da estimativa/realizado feito pela equipe encontra-se detalhado na Tabela 2, que possui quatro colunas, sendo respectivamente:

- Tarefa – Número da requisição de mudança solicitada para o desenvolvimento;

- Tempo Estimado (Horas) – Tempo que a equipe sugeriu, utilizando a técnica *Planning Poker*;
- Tempo Gasto (Horas) – Tempo real gasto para cada requisição de mudança;
- Tipo de Caso (P=Pequeno, M=Médio, G=Grande) – Grandeza estimada pela equipe para cada atividade de mudança do software.

Para esse *Sprint* foram selecionados 4 (quatro) desenvolvedores da equipe, escolhidos com base na experiência do gerente de projetos.

Tabela 2: Requisições de mudança e estimativa de tempo calculado pela equipe da empresa parceira

Sprint verso 3.0.04 Data 11/11/2015				
Dev.	**Tarefa**	**Estimado (H)**	**Gasto (H)**	**Tipo de Caso (PMG)**
Dev. 1	13357	2.5	0.3	P
	13076	2.5	0.6	P
	13427	2.5	0	P
	13346	2.5	10.4	P
	13455	2.5	1.4	P
	13446	2.5	0.2	P
	13441	2.5	0	P
	13372	2.5	5.6	P
	13266	2.5	0	P
	13393	2.5	0	P
	13413	2.5	0	P
	13365	2.5	0	P
	13214	2.5	0	P
	13401	2.5	0	P
Dev. 2	13186	26.3	23.2	M
	13452	2.5	0.3	P
	13436	2.5	0.2	P
	13411	2.5	3	P
	13410	2.5	1.1	P
	13241	2.5	6.7	P
	13169	2.5	1.1	P
	13387	2.5	0.1	P
	13386	2.5	0.6	P
	13049	0	0.1	P
	13358	2.5	0	P
Dev. 3	12693	26.3	24.6	M
	13242	2.5	6.6	P
	13360	2.5	6.7	P
	13348	2.5	6.3	P
	13347	2.5	6.1	P
Dev. 4	12897	0	0.4	M
	12590	26.3	24.4	M
	13437	2.5	1.9	P
	13250	2.5	2	P
	13422	2.5	3.3	P
	13400	2.5	0.4	P
	13398	2.5	0.6	P
	13395	2.5	0.3	P
	13282	26.3	12.2	M
	13381	2.5	0.4	P
TOTAL	**40**	**190.2**	**151.1**	

4.2 MEDIÇÕES

Com as informações indicadas anteriormente, iniciou-se a configuração do *framework GiveMe Effort.* Foram realizadas 4 (quatro) calibrações diferentes para viabilizar a análise e medir os resultados.

1) A primeira configuração dos dados no *framework* encontra-se disposta na Tabela 3.

Tabela 3: Configuração - medição

Configuração		MEDIÇÃO 1
Percentual Similaridade SUMO	50%	Tipo de Análise: Sumário
Distância de Levenshtein	20 Trocas	Anos considerados: 99
Reputação dos Desenvolvedores	40%	Remoção de *Outliers*: NÃO

Após a execução do cálculo de estimativa de esforço, para cada uma das requisições de mudança, apresentada na Tabela 2, uma nova tabela foi gerada com os valores atribuídos a cada uma das técnicas, em função da requisição de mudança calculada. O resultado final, com a comparação das técnicas, é apresentado na Tabela 4.

Tabela 4: Medição 1 – Comparação das Técnicas

Sprint version 3.0.04 Data 11/11/2015									
Dev.	Tarefa	Tempo Estimado (Horas)	Tempo Gasto (Horas)	Tipo de Caso (PMG)	Estimado SUMO	Estimado Levenshtein	Estimado Reputação	Estimado Interseção	Estimado Regressão Linear
Dev. 1	13357	2.5	0.3	P	57	39	101	88	33.25
	13076	2.5	0.6	P	0	0	0	0	0
	13427	2.5	0	P	18	0	26	0	0
	13346	2.5	10.4	P	3	65	143	5	3.34
	13455	2.5	1.4	P	2	0	2	0	0
	13446	2.5	0.2	P	0	0	0	0	0
	13441	2.5	0	P	252	180	84	0	9.17
	13372	2.5	5.6	P	85	2	0	0	0
	13266	2.5	0	P	47	89	6	0	0
	13393	2.5	0	P	0	0	0	0	0
	13413	2.5	0	P	0	0	0	0	0
	13365	2.5	0	P	69	391	143	391	0
	13214	2.5	0	P	484	0	0	0	0
	13401	2.5	0	P	0	0	0	0	0
Dev. 2	13186	26.3	23.2	M	19	0	9	0	0
	13452	2.5	0.3	P	2	242	0	0	10.32
	13436	2.5	0.2	P	2	0	0	0	0.66
	13411	2.5	3	P	1	45	1	0	0
	13410	2.5	1.1	P	2	46	1	0	4.12
	13241	2.5	6.7	P	0	8	0	0	0
	13169	2.5	1.1	P	20	0	0	0	3.09
	13387	2.5	0.1	P	0	211	0	0	0
	13386	2.5	0.6	P	39	82	39	103	17.16
	13049	0	0.1	P	0	0	0	0	0
	13358	2.5	0	P	0	0	0	0	0

Dev. 3	12693	26.3	24.6	M	0	0	0	0	0
	13242	2.5	6.6	P	2	47	1	0	8.6
	13360	2.5	6.7	P	32	8	13	0	13.24
	13348	2.5	6.3	P	33	54	16	0	12.18
	13347	2.5	6.1	P	31	7	13	0	15.44
Dev. 4	12897	0	0.4	M	0	0	0	0	0
	12590	26.3	24.4	M	0	0	0	0	0
	13437	2.5	1.9	P	0	0	0	0	0
	13250	2.5	2	P	1	148	2	0	1
	13422	2.5	3.3	P	0	0	0	0	0
	13400	2.5	0.4	P	0	0	0	0	0
	13398	2.5	0.6	P	0	170	9	0	2.05
	13395	2.5	0.3	P	0	0	0	0	2.03
	13282	26.3	12.2	M	38	21	0	0	2.03
	13381	2.5	0.4	P	199	204	99	1	0
TOTAL	40	190.2	151.1	0	1438	2059	708	588	137.68

A Figura 32 apresenta um gráfico comparativo entre as técnicas. São apresentados o valor calculado e o total de estimativa de esforço para cada técnica, conforme legenda.

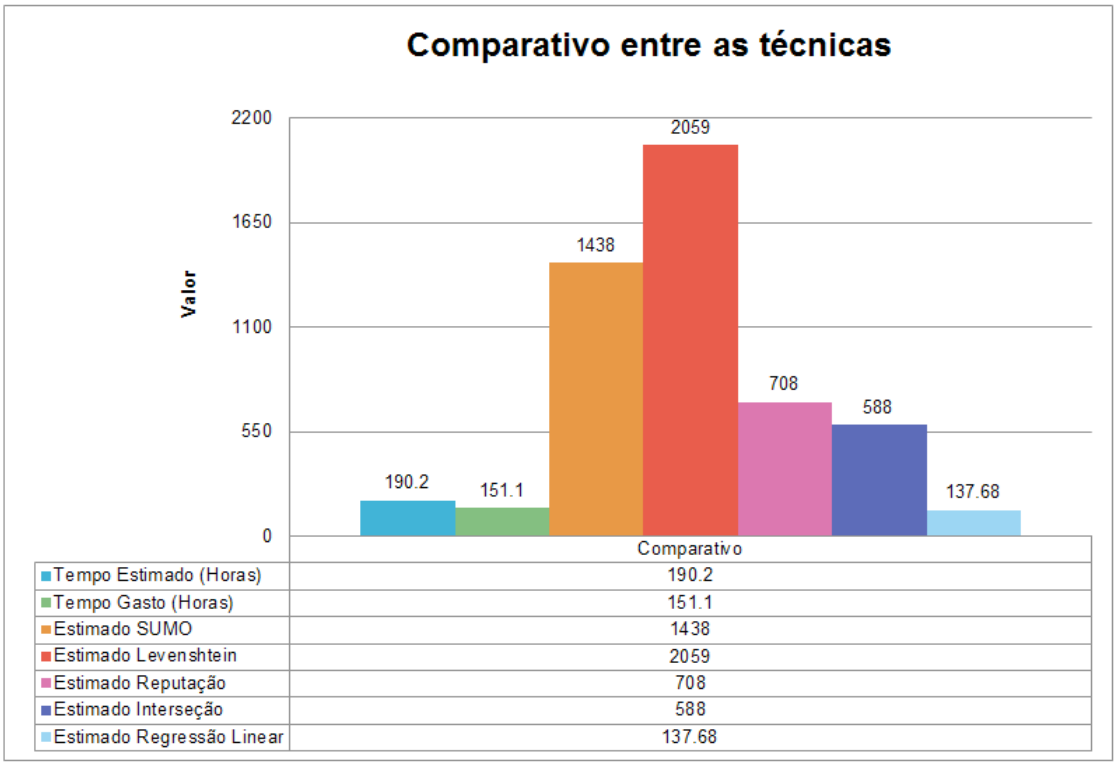


Figura 33: MEDIÇÃO 1 - Gráfico comparativo das técnicas

Pode-se notar que a técnica Levenshtein estimou uma quantidade muito grande de horas, aproximadamente 1907 horas a mais que o real gasto. No entanto, a técnica de regressão linear se aproximou bastante do tempo real gasto, com uma margem de erro de aproximadamente 9,74 %.

2) A segunda configuração no *framework* foi aplicada aos dados demonstrados na Tabela 5.

Tabela 5: Medição 2 – calibração

Configuração		**MEDIÇÃO 2**
Percentual Similaridade SUMO	**40%**	Tipo de Análise: Sumário
Distância de Levenshtein	**10 Trocas**	**Anos considerados: 1**
Reputação dos Desenvolvedores	40%	Remoção de *Outliers*: NÃO

Observa-se que houve uma diminuição no percentual de similaridade da técnica SUMO e foi reduzido o número de trocas para a técnica Distância de Levenshtein. Foram considerados para esse cálculo apenas o último ano de requisições de mudanças. Após a execução do cálculo de estimativa de esforço, uma nova tabela foi gerada com os valores atribuídos a cada uma das técnicas em função da requisição de mudança calculada. O resultado final, com a comparação das técnicas, é apresentado na Tabela 6.

Tabela 6: Medição 2 – Comparação das Técnicas

Sprint version 3.0.04 Data 11/11/2015									
Dev.	Tarefa	Tempo Estimado (Horas)	Tempo Gasto (Horas)	Tipo de Caso (PMG)	Estimado SUMO	Estimado Levenshtein	Estimado Reputação	Estimado Interseção	Estimado Regressão Linear
Dev. 1	13357	2.5	0.3	P	19	38	36	38	0.57
	13076	2.5	0.6	P	0	0	0	0	0
	13427	2.5	0	P	4	0	5	0	0
	13346	2.5	10.4	P	4	0	12	0	5
	13455	2.5	1.4	P	2	0	2	0	0
	13446	2.5	0.2	P	0	0	0	0	0
	13441	2.5	0	P	21	0	2	0	0
	13372	2.5	5.6	P	9	0	0	0	0
	13266	2.5	0	P	26	0	6	0	0
	13393	2.5	0	P	0	0	0	0	0
	13413	2.5	0	P	8	0	3	0	2
	13365	2.5	0	P	20	0	21	0	39.96
	13214	2.5	0	P	4	0	0	0	0
	13401	2.5	0	P	0	0	0	0	0
Dev. 2	13186	26.3	23.2	M	18	0	15	0	35.64
	13452	2.5	0.3	P	4	2	0	0	2
	13436	2.5	0.2	P	1	0	1	0	0
	13411	2.5	3	P	3	1	0	0	0
	13410	2.5	1.1	P	3	3	0	0	1.49
	13241	2.5	6.7	P	0	0	0	0	0
	13169	2.5	1.1	P	54	0	53	0	5.36
	13387	2.5	0.1	P	3	0	0	0	0
	13386	2.5	0.6	P	17	0	14	0	51.78
	13049	0	0.1	P	0	0	0	0	0
	13358	2.5	0	P	0	0	0	0	0
Dev. 3	12693	26.3	24.6	M	0	0	0	0	0
	13242	2.5	6.6	P	3	0	10	0	3
	13360	2.5	6.7	P	19	0	14	0	24.05
	13348	2.5	6.3	P	20	10	15	0	54.58
	13347	2.5	6.1	P	19	10	14	0	58.42

Dev. 4	12897	0	0.4	M	0	0	0	0	0
	12590	26.3	24.4	M	3	0	1	0	0.8
	13437	2.5	1.9	P	1	2	2	0	0
	13250	2.5	2	P	16	0	23	0	0.5
	13422	2.5	3.3	P	20	0	0	0	0
	13400	2.5	0.4	P	26	0	0	0	0
	13398	2.5	0.6	P	1	0	0	0	0
	13395	2.5	0.3	P	2	0	0	0	0
	13282	26.3	12.2	M	30	0	0	0	0
	13381	2.5	0.4	P	126	3	1	1	1.26
TOTAL	40	190.2	151.1	0	506	69	250	39	286.41

O gráfico da Figura 34 apresenta os valores totais de cada uma das técnicas geradas pelo *GiveMe Effort* para a segunda medição dos dados.

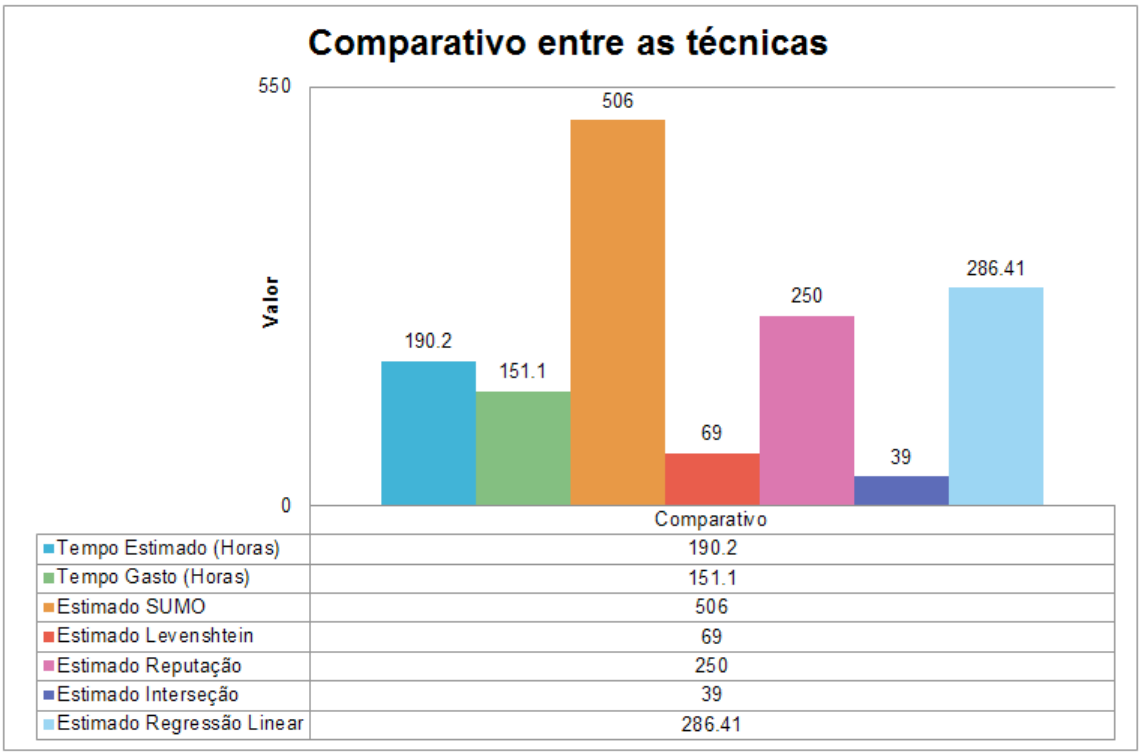


Figura 34: MEDIÇÃO 2 - Gráfico comparativo das técnicas

É possível observar que, à medida que se diminui a quantidade de trocas na configuração da técnica Levenshtein, o tempo estimado pela mesma decresce sensivelmente, ainda assim, apresenta-se bem distante do real gasto. A média geral estimada pela ferramenta foi de 230,08 horas, aproximadamente 20,97% a mais que o tempo estimado pela equipe de desenvolvimento. A técnica SUMO, para esse tipo de configuração, apresentou um gasto de 354,9 horas a mais que o real gasto. Já a técnica de Regressão Linear aumentou sensivelmente em relação à medição anterior, ficando em aproximadamente 51%, a mais do que a estimativa feita pela equipe de desenvolvimento.

3) A terceira configuração foi aplicada aos dados dispostos na Tabela 7. A diferença dessa configuração para a anterior está na inclusão da remoção automática de *outliers* no cálculo de todas as técnicas.

Tabela 7: Medição 3 – calibração

Configuração		**MEDIÇÃO 3**
Percentual Similaridade SUMO	40%	Tipo de Análise: Sumário
Distância de Levenshtein	10 Trocas	Anos considerados: 1
Reputação dos Desenvolvedores	40%	**Remoção de *Outliers*: SIM**

Assim como realizado nas medições anteriores, uma nova tabela foi gerada com os valores atribuídos a cada uma das técnicas em função da requisição de mudança calculada. O resultado final, com a comparação das técnicas, é apresentando na Tabela 8, a qual é apresentada a seguir.

Tabela 8: Medição 3 – comparação das técnicas

Sprint version 3.0.04 Data 11/11/2015									
Dev.	Tarefa	Tempo Estimado (Horas)	Tempo Gasto (Horas)	Tipo de Caso (PMG)	Estimado SUMO	Estimado Levenshtein	Estimado Reputação	Estimado Interseção	Estimado Regressão Linear
Dev. 1	13357	2.5	0.3	P	2	38	16	2	0.57
	13076	2.5	0.6	P	0	0	0	0	0
	13427	2.5	0	P	7	0	7	0	0
	13346	2.5	10.4	P	2	0	5	0	5
	13455	2.5	1.4	P	0	0	0	0	0
	13446	2.5	0.2	P	0	0	0	0	0
	13441	2.5	0	P	21	0	2	0	0
	13372	2.5	5.6	P	6	0	0	0	0
	13266	2.5	0	P	6	0	6	0	0
	13393	2.5	0	P	0	0	0	0	0
	13413	2.5	0	P	3	0	3	0	2
	13365	2.5	0	P	4	0	4	0	39.96
	13214	2.5	0	P	6	0	1	0	0
	13401	2.5	0	P	0	0	0	0	0
Dev. 2	13186	26.3	23.2	M	3	0	3	0	35.64
	13452	2.5	0.3	P	2	2	0	2	2
	13436	2.5	0.2	P	1	0	1	0	0
	13411	2.5	3	P	3	0	0	0	0
	13410	2.5	1.1	P	3	0	0	0	1.49
	13241	2.5	6.7	P	0	0	0	0	0
	13169	2.5	1.1	P	19	0	16	0	5.36
	13387	2.5	0.1	P	3	0	0	0	0
	13386	2.5	0.6	P	3	0	2	0	51.78
	13049	0	0.1	P	0	0	0	0	0
	13358	2.5	0	P	0	0	0	0	0
Dev. 3	12693	26.3	24.6	M	0	0	0	0	0
	13242	2.5	6.6	P	0	0	0	0	3
	13360	2.5	6.7	P	2	0	2	0	24.05
	13348	2.5	6.3	P	2	0	2	0	54.58
	13347	2.5	6.1	P	2	0	2	0	58.42
Dev. 4	12897	0	0.4	M	0	0	0	0	0
	12590	26.3	24.4	M	1	0	1	0	0.8
	13437	2.5	1.9	P	1	0	2	0	0
	13250	2.5	2	P	1	0	2	0	0.5

	13422	2.5	3.3	P	5	0	0	0	0
	13400	2.5	0.4	P	0	0	0	0	0
	13398	2.5	0.6	P	0	0	0	0	0
	13395	2.5	0.3	P	2	0	0	0	0
	13282	26.3	12.2	M	30	0	0	0	0
	13381	2.5	0.4	P	8	3	1	1	1.26
TOTAL	40	190.2	151.1	0	148	43	78	5	286.41

A Figura 35 apresenta uma análise gráfica de comparação para as técnicas aplicadas na terceira medição de dados.

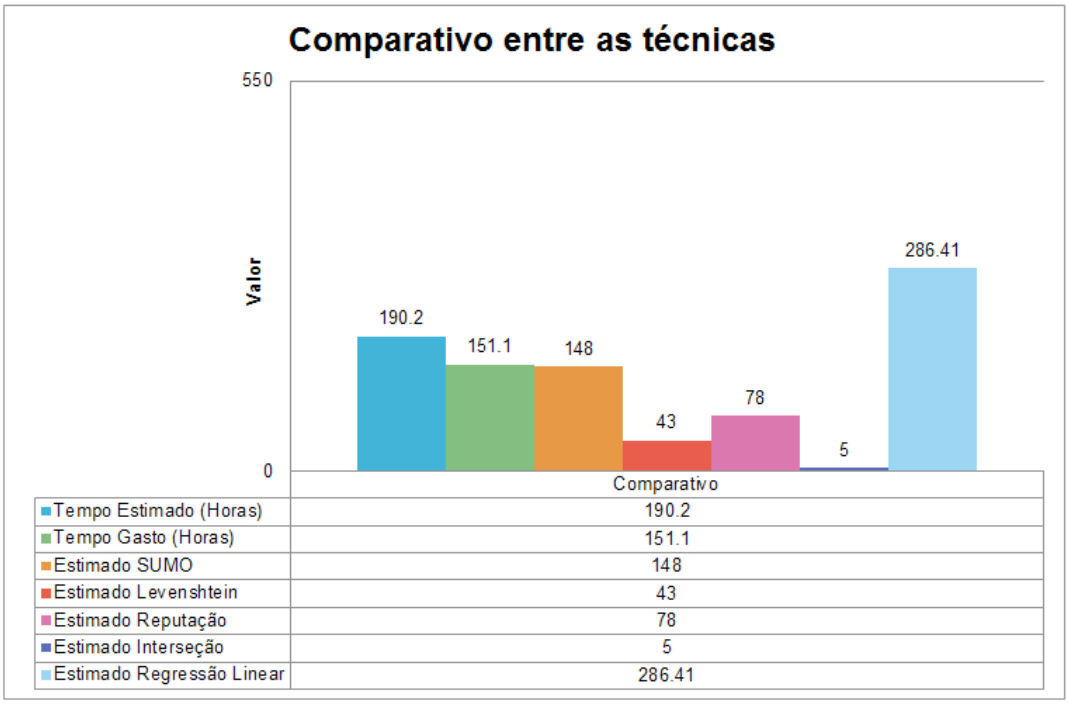


Figura 35: MEDIÇÃO 3 - Gráfico comparativo das técnicas

Nota-se que, com a remoção dos *outliers* na configuração padrão, houve uma diminuição no tempo estimado pelas técnicas (SUMO, Levenshtein e Reputação), no entanto a estimativa por regressão linear se manteve inalterada. A média geral estimada foi de 112,08 horas, aproximadamente 34,81%, a menos que o tempo real gasto. E possível notar que a Técnica Interseção diminui sensivelmente, indicando que as requisições de mudanças encontradas por essa técnica possivelmente eram *outliers*.

4) A quarta configuração no *framework* foi aplicada aos dados, conforme demonstrado na Tabela 9. A diferença dessa configuração em relação a

anterior está no filtro dos anos considerados para a análise das requisições. Foi incluído apenas 1 (um) ano histórico de requisições para a análise das técnicas. O percentual de similaridade para a técnica SUMO foi aumentado para 50%, denotando uma maior similaridade textual entre a requisição solicitada e a base histórica. A distância de Levenshtein foi alterada para 20 trocas, permitindo assim, um número maior de requisições de mudanças no espectro de busca da ferramenta.

Tabela 9: Medição 4 – calibração da ferramenta

Configuração		MEDIÇÃO 4
Percentual Similaridade SUMO	**50%**	Tipo de Análise: Sumário
Distância de Levenshtein	**20 Trocas**	Anos Considerados: **1**
Reputação dos Desenvolvedores	40%	Remoção de *Outliers*: SIM

Assim como nas demais medições, após a execução do cálculo de estimativa de esforço pelo *framework GiveMe Effort,* foi gerada uma nova tabela com a comparação das técnicas, apresentada na Tabela 10, a seguir.

Tabela 10: Medição 4 – comparação das técnicas

Sprint version 3.0.04 Data 11/11/2015									
Dev.	Tarefa	Tempo Estimado (Horas)	Tempo Gasto (Horas)	Tipo de Caso (PMG)	Estimado SUMO	Estimado Levenshtein	Estimado Reputação	Estimado Interseção	Estimado Regressão Linear
Dev. 1	13357	2.5	0.3	P	1	2	1	2	33
	13076	2.5	0.6	P	0	0	0	0	0
	13427	2.5	0	P	3	0	4	0	0
	13346	2.5	10.4	P	3	8	12	5	3.3
	13455	2.5	1.4	P	0	0	0	0	0
	13446	2.5	0.2	P	0	0	0	0	0
	13441	2.5	0	P	2	0	2	0	9.1
	13372	2.5	5.6	P	0	0	0	0	0
	13266	2.5	0	P	6	0	6	0	0
	13393	2.5	0	P	0	0	0	0	0
	13413	2.5	0	P	3	0	3	0	0
	13365	2.5	0	P	3	0	2	0	0
	13214	2.5	0	P	4	0	0	0	0
	13401	2.5	0	P	0	0	0	0	0
Dev. 2	13186	26.3	23.2	M	3	0	3	0	10
	13452	2.5	0.3	P	2	4	0	2	5.3
	13436	2.5	0.2	P	3	8	12	5	0.6
	13411	2.5	3	P	1	1	1	0	0
	13410	2.5	1.1	P	2	1	1	0	4.1
	13241	2.5	6.7	P	0	0	0	0	0
	13169	2.5	1.1	P	12	0	0	0	3.09
	13387	2.5	0.1	P	0	0	0	0	0
	13386	2.5	0.6	P	3	3	2	2	17.16
	13049	0	0.1	P	0	0	0	0	0

	13358	2.5	0	P	0	0	0	0	0
Dev. 3	12693	26.3	24.6	M	0	0	0	0	0
	13242	2.5	6.6	P	2	4	1	0	8.6
	13360	2.5	6.7	P	6	6	9	0	13.24
	13348	2.5	6.3	P	4	7	2	0	12.18
	13347	2.5	6.1	P	4	5	2	0	15.44
Dev. 4	12897	0	0.4	M	0	0	0	0	0
	12590	26.3	24.4	M	0	0	0	0	0
	13437	2.5	1.9	P	1	0	2	0	0
	13250	2.5	2	P	1	0	2	0	1
	13422	2.5	3.3	P	0	0	0	0	0
	13400	2.5	0.4	P	0	0	0	0	0
	13398	2.5	0.6	P	0	4	5	0	2.05
	13395	2.5	0.3	P	0	0	0	0	0
	13282	26.3	12.2	M	38	21	0	30	2.3
	13381	2.5	0.4	P	3	0	0	0	15.04
TOTAL	40	190.2	151.1	0	110	74	72	46	154.9

O gráfico da Figura 36 apresenta os valores totais de cada uma das técnicas geradas pelo *GiveMe Effort* para a quarta medição de dados.

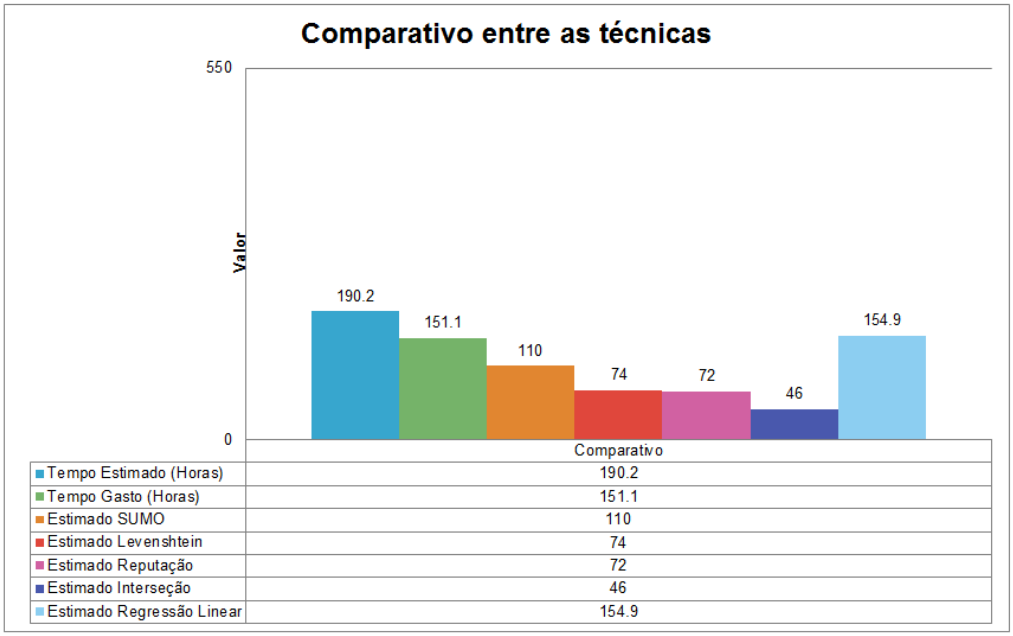


Figura 36: MEDIÇÃO 4 - Gráfico comparativo das técnicas

É possível observar que, como a remoção dos *outliers* estava ativa, a quantidade de horas apresentada, nas técnicas (SUMO, Levenshtein e Reputação), continuou abaixo do tempo estimado pelos desenvolvedores e o tempo real gasto. A média geral estimada pelo *framework* foi de 91,38 horas, aproximadamente 65,35%, a menos que o tempo real gasto. No entanto, a técnica de regressão linear obteve uma margem de erro de aproximadamente 2%, em relação ao real gasto e uma margem de 22,78%

aproximadamente para o tempo estimado pelos desenvolvedores. Pode-se notar também, um aumento na estimativa da Técnica Intersection, em relação a medição anterior, indicando possivelmente que a busca por requisições parecidas obteve mais resultados devido as calibrações configuradas nas técnicas SUMO e Levenshtein.

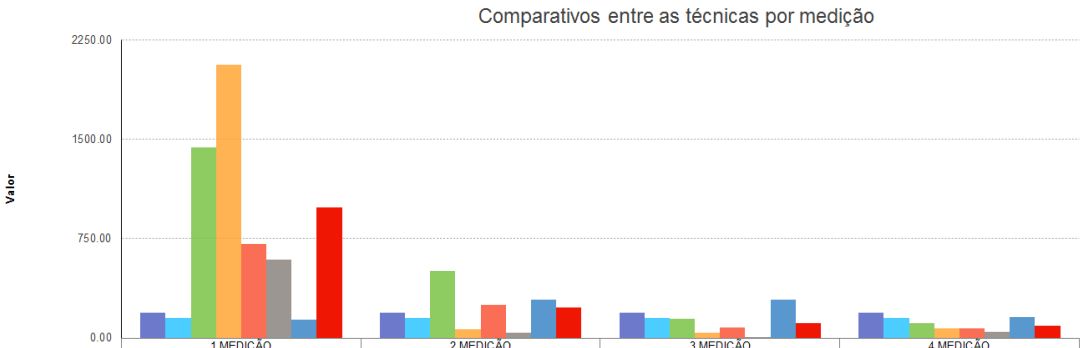
A Tabela 11 faz uma comparação entre os valores apresentados em todas as medições. Pode-se observar na coluna Valores Fixos os valores Estimado e Gasto pela equipe de desenvolvimento da empresa parceira em um *Sprint* realizado. As quatro medições são apresentas em suas colunas respectivas, bem como, as estimativas de esforço encontradas em cada técnica. Ao final, a tabela apresenta a média aritmética das técnicas em cada medição. A primeira medição foi realizada com uma calibração que elevou a quantidade de horas da estimativa ficando sua média muito discrepante dos valores esperados. Essa calibração foi feita propositalmente para evidenciar a necessidade de um especialista, e para o caso em que o *framework*, e suas técnicas não sejam corretamente calibrados e entendidos, pode-se gerar valores que não atendam a equipes em sua estimativa de esforço. A segunda medição apresentou aproximadamente 21,08% a mais que o tempo estimado pela equipe. A terceira medição foi 69,00% a menor que o estimado pela equipe, e a quarta medição foi 79,44% menor que o estimado.

Tabela 11: Comparação das técnicas em função das medições

Valores Fixos	Estimado PMG - Equipe		190.02	
	Tempo Gasto - Equipe		151.1	
DESCRIÇÃO	1 MEDIÇÃO	2 MEDIÇÃO	3 MEDIÇÃO	4 MEDIÇÃO
Sumo	1438.00	506.00	148.00	110.00
Levenshtein	2059.00	69.00	43.00	74.00
Reputação	708.00	250.00	78.00	72.00
Interseção	588.00	39.00	5.00	46.00
Regressão Linear	137.68	286.41	286.41	154.90
Média das Técnicas	986.14	230.08	112.08	91.38

A figura 37 apresenta um comparativo entre as técnicas em cada uma das medições, incluindo o tempo Estimado e Gasto para a comparação. Cada técnica foi apresentada no gráfico em cada uma das medições, e possibilita a comparação das mesmas e seu comportamento à medida que se calibra o framework de forma diferente. Alguns destaques podem ser notados como os valores obtidos pela regressão linear na primeira e quarta medições que apresentaram valores bem próximos ao real gasto pela equipe. Em média, a segunda calibração se aproximou mais do valor estimado pela

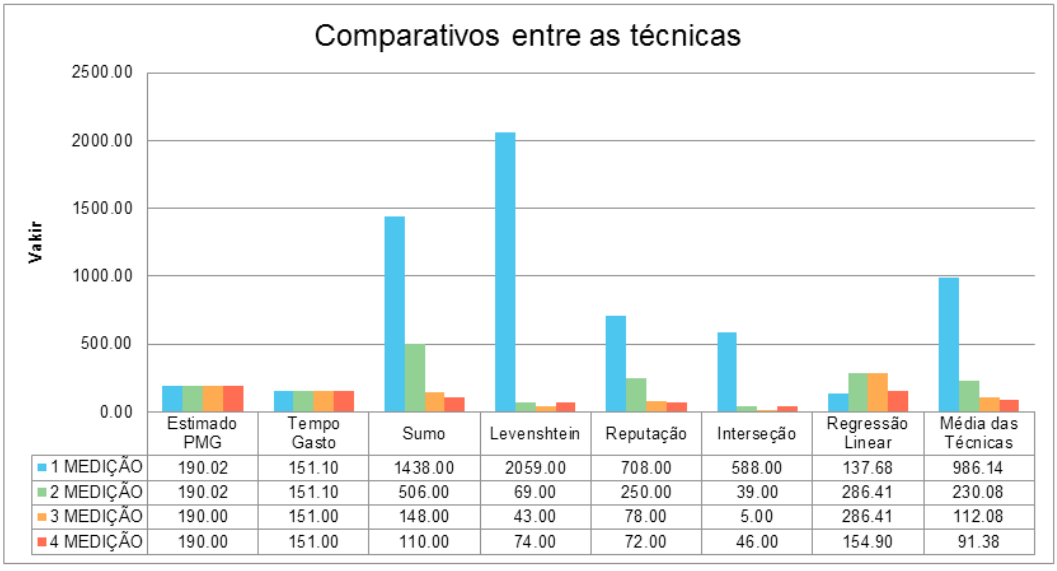
equipe.



	1 MEDIÇÃO	2 MEDIÇÃO	3 MEDIÇÃO	4 MEDIÇÃO
Estimado PMG	190.02	190.02	190.00	190.00
Tempo Gasto	151.10	151.10	151.00	151.00
Sumo	1438.00	506.00	148.00	110.00
Levenshtein	2059.00	69.00	43.00	74.00
Reputação	708.00	250.00	78.00	72.00
Interseção	588.00	39.00	5.00	46.00
Regressão Linear	137.68	286.41	286.41	154.90
Média das Técnicas	986.14	230.08	112.08	91.38

Figura 37: Comparativo entre as medições realizadas

A figura 38 apresenta um comparativo entre as cada uma das técnicas em função das medições. Podem-se observar todas as técnicas e em cada medição seu comportamento. Pode-se notar que a Regressão Linear manteve um comportamento equilibrado em relação as demais técnicas e que a Técnica Levenshtein sofreu picos na primeira medição e, à medida que foi calibrada corretamente (menor índice de trocas) ela se manteve mais equilibrada com as demais técnicas.



	Estimado PMG	Tempo Gasto	Sumo	Levenshtein	Reputação	Interseção	Regressão Linear	Média das Técnicas
1 MEDIÇÃO	190.02	151.10	1438.00	2059.00	708.00	588.00	137.68	986.14
2 MEDIÇÃO	190.02	151.10	506.00	69.00	250.00	39.00	286.41	230.08
3 MEDIÇÃO	190.00	151.00	148.00	43.00	78.00	5.00	286.41	112.08
4 MEDIÇÃO	190.00	151.00	110.00	74.00	72.00	46.00	154.90	91.38

Figura 38: Comparativo entre as técnicas

A figura 39 apresenta um gráfico de área para um comparativo entre as técnicas e suas medições ao longo do tempo. Pode se perceber nitidamente os picos e diferenças da técnica Levenshtein, que se mal calibrada, apresenta medições muito discrepantes do

real, e isso se deve a natureza da técnica, visto que um baixo índice de trocas indica uma proximidade maior entre as requisições de mudança encontrada e estimada.

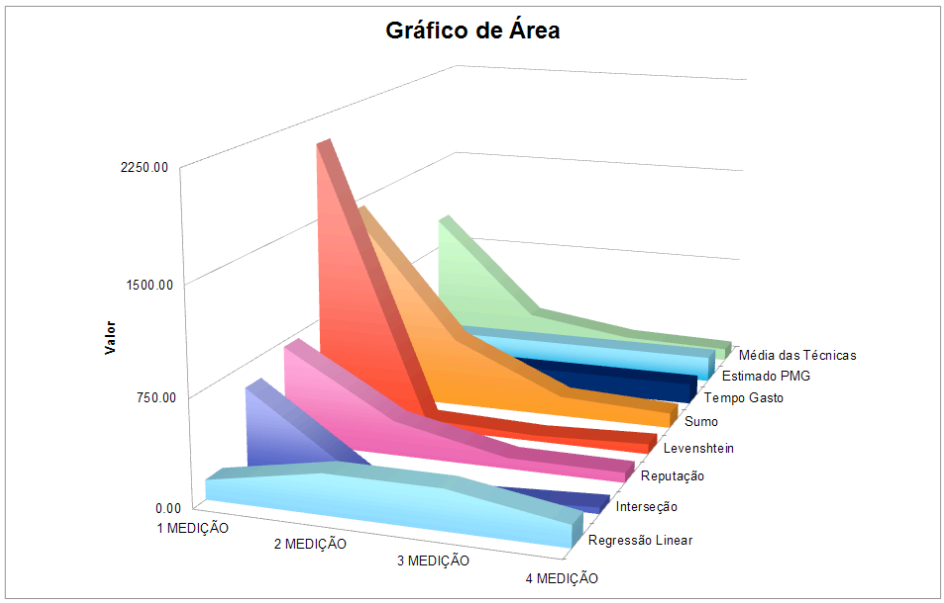


Figura 39: Gráfico de área comparativo entre as medições realizadas

4.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

O objetivo dessa seção foi demonstrar a utilização da ferramenta em um contexto real de uma empresa de desenvolvimento de software, bem como a possibilidade de parametrização do *framework*. As calibrações realizadas no *framework* foram comparadas aos valores Previsto e Realizado efetuado pela equipe da empresa parceira no *Sprint* selecionado. As análises permitiram notar que é possível configurar o *GiveMe Effort* de diversas formas diferentes, obtendo valores de estimativa de esforço que precisam ser analisados. Logo após, é possível selecionar a calibração que seja mais aderente à equipe de desenvolvimento.

Pode-se notar que as médias das técnicas apresentadas nas três últimas medições se mostraram aproximadas ao tempo real gasto e estimado pela empresa. O destaque fica para a segunda medição que se aproximou mais, com sua calibração, do valor estimado e gasto pela equipe. A primeira medição mostrou valores muito discrepantes, em média, devido à natureza da configuração realizada. Isso evidencia que se mal calibrado o *framework* pode apresentar valores muito diferentes do esperado, e também que, para a calibração inicial, existe a necessidade de um especialista para que se possa comparar os resultados apresentados pela calibração e os resultados esperados pela equipe.

A próxima seção apresentará à avaliação do *framework* por meio de um experimento realizado, sendo dispostas também as ameaças à validade do mesmo.

5. AVALIAÇÃO DA SOLUÇÃO

Com o intuito de verificar a viabilidade de utilização do *GiveMe Effort*, um estudo experimental foi executado. Foi conduzido em um contexto real de utilização, com os dados e os desenvolvedores de uns dos projetos da empresa parceira. Além disso, um gerente de projetos externo à empresa foi convidado a participar do experimento. O objeto do convite do gerente externo foi para que pudéssemos obter uma análise imparcial, bem como, identificar como o *framework* pode auxiliar na estimativa de esforço, mesmo que não se conheça a regra de negócio da empresa ou se tenha qualquer tipo de contato com a base histórica da mesma. Isso se faz necessário, pois muitas vezes, no dia a dia de uma empresa de desenvolvimento de software precisa-se contratar um novo desenvolvedor ou um novo gerente para a equipe. Como resultado, a curva de aprendizado dos mesmos, para que eles possam mensurar ou acompanhar as estimativas de esforço, de forma mais assertiva, é grande. Com isso, tentou-se mostrar o valor do *framework* para apoiar a estimativa de esforço em atividades de manutenção e compreensão de software.

5.1 PLANEJAMENTO DO ESTUDO EXPERIMENTAL

Seguindo o modelo GQM (*Goal/Question/Metric*) (BASILI, 1994) foram definidos os objetivos da prova de conceito apresentada e, em seguida, as questões de pesquisa, juntamente à hipótese. Posteriormente, métricas foram definidas para cada uma das questões de pesquisa, tal como pode ser visto na Figura 40.

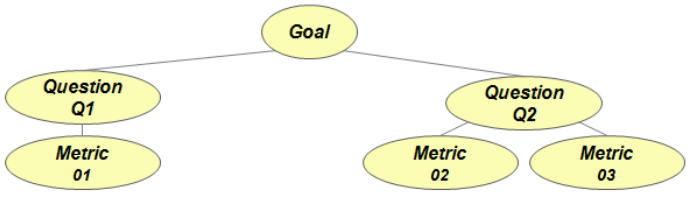


Figura 40: Hierarquia do modelo GQM.

Os parâmetros definidos para o respectivo modelo encontram-se dispostos na sequência:

Goal (Objetivo): Analisar o *framework GiveMe Effort* com a finalidade de verificar a viabilidade de cálculo da estimativa de esforço em projetos de software, no contexto de desenvolvimento Ágil, e a contribuição deste na compreensão e manutenção de softwares.

Questions (Questões):

- Q1: Como o *GiveMe Effort* proporciona a equipe de desenvolvimento mecanismos que possibilitam o cálculo da estimativa de esforço nas atividades manutenção de software?
- Q2: Como o *GiveMe Effort* apoia a manutenção e evolução de softwares através de visualizações?

Com base nos objetivos, foram definidas as seguintes hipóteses:

- H_{nula}: a utilização do *GiveMe Effort* **não** é capaz de apoiar a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis em um ambiente distribuído de software;

- $H_{alternativa}$: a utilização do *GiveMe Effort* **é** capaz de apoiar a estimativa de esforço nas atividades de manutenção e compreensão de software em um AIMV, em projetos ágeis em um ambiente distribuído de software.

- *Metrics* (Métricas): foram definidas com base nas questões de pesquisa Q1 e Q2, as seguintes métricas, sendo a Métrica 1 e 2 referentes a questão 1, e a Métrica 3 a questão 2:
    - M1: percentual de proximidade do resultado do cálculo da estimativa de esforço em relação ao gasto real, utilizando o *framework GiveMe Effort*;
    - M2: percentual de proximidade do resultado do cálculo da estimativa de esforço em relação ao previsto pela equipe, sem utilização do *framework GiveMe Effort*;
    - M3: quantidade de visualizações efetuadas para auxiliar a equipe de desenvolvimento no cálculo da estimativa de esforço, utilizando o *framework GiveMe Effort*.

A próxima subseção demonstrará o estudo experimental conduzido para o *framework GiveMe Effort*.

5.2 CONDUÇÃO DO ESTUDO EXPERIMENTAL

Para a condução do estudo experimental foi realizado um planejamento prévio que envolveu a construção de um Site[5] e, no mesmo, foi incluído o material preparado para o treinamento, tutorial de uso do *framework*, artigos publicados, formulários de caracterização e consentimento, bem como, os resultados obtidos pelo uso do *framework* por cada participante.

Após esse passo foi realizado um treinamento *online* com o todos os participantes, visto que os mesmos estavam geograficamente distribuídos. Seguido o treinamento, foram conduzidos os passos nas subseções a seguir.

5.3 TERMO DE CONSENTIMENTO DOS PARTICIPANTES

Para a realização do estudo experimental foi criado um Termo de Consentimento[3] que definiu o procedimento, tratamento de possíveis riscos e desconfortos, benefícios e custos, confidencialidade da pesquisa, termo de participação e por fim uma Declaração de Consentimento.

5.4 CARACTERIZAÇÃO DOS PARTICIPANTES

Para a realização do estudo experimental também foi realizado um processo de Caracterização do Experimento[3] que visou identificar alguns dados a respeito dos participantes como nome, idade, sexo, profissão, escolaridade, bem como, responder as seguintes perguntas:

a) Como você avalia seu conhecimento sobre Estimativa de Esforço?

---

[5] GiveMe Effort - Disponível em: http://givemeeffort.tk. Acesso em 01 ago. 2016.

b) Cite as ferramentas/*frameworks* de avaliação de Estimativa de Esforço que já utilizou?

c) Como você avalia seu conhecimento sobre atividades de manutenção e compreensão de software?

d) Cite as ferramentas/*frameworks* de avaliação de manutenção e compreensão de software que já utilizou?

Após a coleta de dados dos 6 (seis) participantes do experimento as seguintes informações foram geradas.

a) Idade dos participantes: Os participantes tinham entre 24 a 40 anos, conforme o gráfico apresentado abaixo (Figura 41).

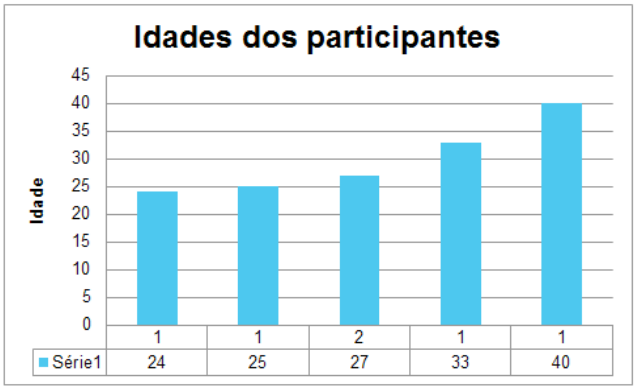


Figura 41: Idade dos participantes

b) Sexo: todos os participantes eram do sexo masculino, conforme a coleta de dados realizada.

c) Profissão: em sua maioria os participantes eram Analistas de Sistemas, sendo que um deles era um Analista de Qualidade e um Coordenador de Desenvolvimento de Projetos, conforme se pode observar na Figura 42. Vale ressaltar, que os todos os profissionais envolvidos, exceto o Coordenador de Desenvolvimento, tinham conhecimento do projeto em andamento e trabalhavam na empresa parceira que forneceu os dados. O Coordenador de Desenvolvimento não tinha qualquer contato com a base de dados de requisições

de mudanças ou qualquer conhecimento das técnicas internas, utilizadas pela empresa parceira, para mensuração de esforço.

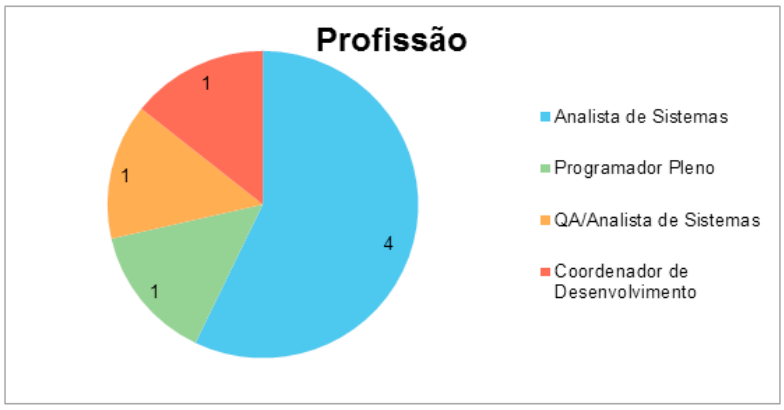


Figura 42: Profissão dos participantes

d) Escolaridade: três dos participantes possuíam Graduação incompleta, dois deles graduação completa e um deles mestrado, conforme pode-se observar na Figura 43. Vale ressaltar que todos os envolvidos no experimento, já trabalham profissionalmente na área de desenvolvimento de software.

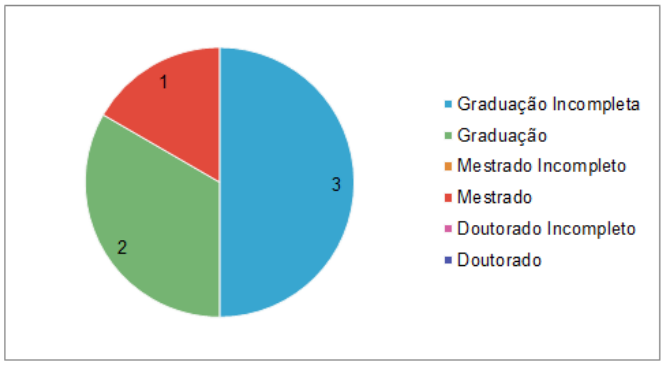


Figura 43: Escolaridade dos participantes

e) Curso: os cursos em que os participantes ingressaram (Figura 44) ou concluíram estão na ara de computação e abrangem de alguma forma a Engenharia de Software com parte de sua grade curricular, o que contribui para o entendimento do valor da pesquisa proposta por esse trabalho.

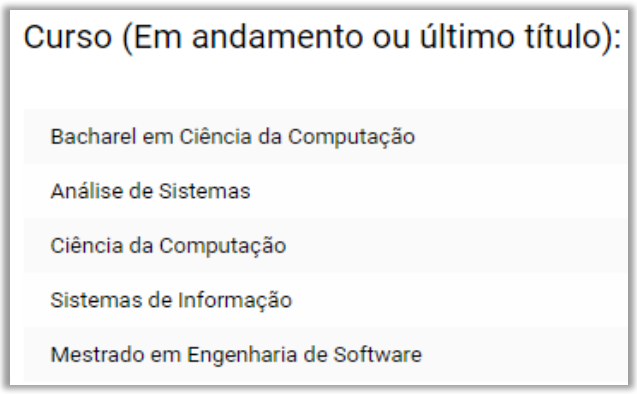


Figura 44: Cursos dos participantes

f) Como você avalia seu conhecimento sobre Estimativa de Esforço? Conforme se pode observar no gráfico apresentando na Figura 45, 50% dos participantes avaliam e possuem um conhecimento regular a respeito de estimativa de esforço, sendo que 2 afirmam ter pouco conhecimento nessa área e um deles um alto conhecimento sobre o tema. Essa pluralidade ajuda a identificar o valor que o *framework* pode agregar mesmo para equipes com pouco ou nenhum conhecimento a respeito de técnicas de estimativa de esforço.

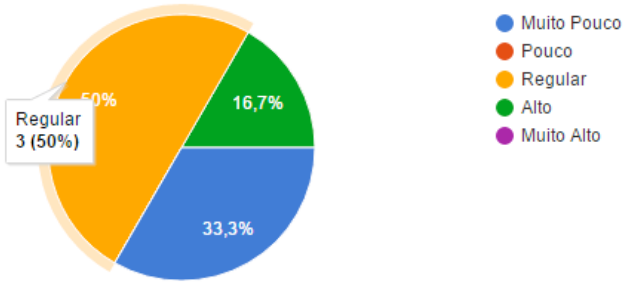


Figura 45: Conhecimento sobre Estimativa de Esforço

g) Cite as ferramentas/*frameworks* de avaliação de Estimativa de Esforço que já utilizou? Segundo as respostas obtidas pelos participantes, foram identificadas ferramentas/*frameworks* para a estimativa de esforço que foram utilizadas por alguns dos membros da pesquisa tais como: TFS Microsoft, SCRUM Hall, *Planning Poker*, Pontos de Função.

h) Como você avalia seu conhecimento sobre atividades de manutenção e compreensão de software? Conforme o gráfico apresentado na Figura 46, temos dois participantes com alto conhecimento sobre Manutenção e Compreensão de Software, dois com conhecimento regular e dois com muito pouco conhecimento nessa área. Essa análise reforça, mais uma vez, o valor do *framework* em ajudar pessoas com pouco ou nenhum conhecimento em atividades de Manutenção e Compreensão de Software, a utilizarem-na no auxilio na Estimativa de Esforço.

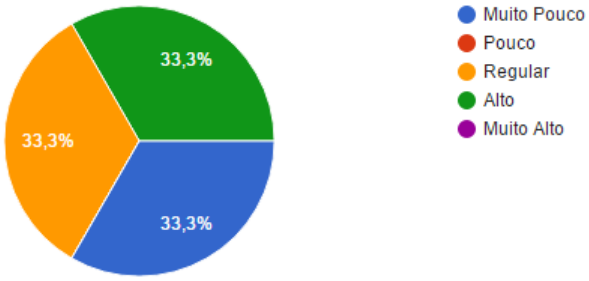


Figura 46: Conhecimento sobre atividades de manutenção e compreensão de software

i) Cite as ferramentas/*frameworks* de avaliação de manutenção e compreensão de software que já utilizou? Por fim se avaliou as ferramentas/*frameworks* de Manutenção e Compreensão de Software utilizadas pelos participantes, para que se pudesse analisar o nível de inovação proposto pelo *GiveMe Effort* em comparação com as que os participantes conheciam. Foram apontas apenas duas ferramentas: SDMetrics[6] e Sonar[7], ambas as ferramentas não abrangem a proposta apresentada no presente trabalho.

Após todo o processo de caracterização deu-se inicio ao experimento com os participantes envolvidos. A próxima subseção irá tratar de mais detalhes a respeito da condução do experimento e os resultados obtidos pelo mesmo.

---

[6] SDMetrics – Disponível em: http://www.sdmetrics.com. Acesso em 01 ago. 2016.

[7] Sonar - Disponível em: http://www.sonarqube.org/. Acesso em 01 ago. 2016.

5.5 ESTUDO EXPERIMENTAL

A condução do Estudo Experimental se deu de forma geograficamente distribuída e agendada, visto que os participantes não se encontravam fisicamente em um mesmo lugar. Após o treinamento coletivo, conforme já detalhado, foi feito um agendamento com cada um dos participantes para que os mesmos podem realizar o experimento. Cada participante recebeu uma planilha (Tabela 12) contendo as requisições de mudança que deveriam analisar no *framework*. Essas requisições fizeram parte de um *Sprint* real realizado pela empresa parceira, e em seguida informavam na mesma, os valores obtidos de estimativa de esforço em cada técnica, bem como avaliaram de forma qualitativa (Tabela 13) cada técnica e visualização existente no *framework*.

Foi solicitado que anotassem as configurações de calibração que utilizaram para as medições, tempo inicial e final, bem como observações pertinentes ao experimento e ao *framework*. As planilhas completas com todos os detalhes de cada coleta de informação, por participante, encontra-se disponível no site do *framework GiveMe Effort* [8], na seção Resultados.

Tabela 12: Requisições de mudança e valores encontrados ao utilizar o *framework*

TEMPO				HORA INICIAL: 8:00 H				HORA FINAL: 10:30 H	
TAREFA	TIPO DE CASO (PMG)	TEMPO GASTO (HORAS)	TEMPO ESTIMADO (HORAS)	TEMPO ESTIMADO SUMO METRICS	ESCALA DE 1 A 5 COMO A TÉCNICA SUMO METRICS APOIOU A ESTIMATIVA DE ESFORÇO	TEMPO ESTIMADO LEVENSHTEIN	ESCALA DE 1 A 5 COMO A TÉCNICA LEVENSHTEIN APOIOU A ESTIMATIVA DE ESFORÇO	TEMPO ESTIMADO REPUTATION	ESCALA DE 1 COMO A TÉCI REPUTATIC APOIOU A ESTIMATIVA ESFORÇO
0014393	M	11	21,4	17	5		5	0	3
0014454	P	0,7	2,2	4	5	0	3	0	3
0014301	P	0,6	2,2	2,2	4	0	3	2,2	4
0014080	M	22	21,4	64	4	0	3	19	5
0014378	P	5	2,2	1,64	4	0	3	2,45	4
0014312	P	2,2	2,2	1,52	4	2,23	4	2,44	4
14348	P	1,4	2,2	1,4	4	6	5	5,5	4
	0	207	264,4	388,2	3,91525423728814	117,35	3,40677966101695	283,41	3,6779661016

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	45%	Tipo de Analise: Sumario
Distância de Levenshtein	20 Trocas	Anos Considerados: 99
Reputação dos Desenvolvedores	10%	Remoção de OutLiers: NAO
Observação Geral a Respeito do Experimento		
Trabalho muito bom e pratico, Todos os gráficos ficaram interessantes, Segue alguns detalhes: Code Analyser me ajudou a ver só os outliers, Reputation tava fixo, gráfico da regressão linear me confundiu com o valor que foi estimado, Box plot ajudou muito na estimativa, Diagrama de Veen não exibe intercessão entre 2 conjuntos, Fixar a configuração para todos os casos nem sempre trouxe resultado,		

Legenda da Escala	
1	Não Apoia a Estimativa de Esforço
2	Pouco Apoiou a Estimativa de Esforço
3	Indiferente
4	Apoia a Estimativa de Esforço
5	Muito Apoiou a Estimativa de Esforço

[8] GiveMe Effort - Disponível em: http://givemeeffort.tk. Acesso em 01 ago. 2016.

Tabela 13: Requisições de mudança e as notas das avaliações qualitativas ao utilizar o *framework*

TAREFA	VIEWS	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO PIE CHART VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO VEEN DIAGRAM VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO BULLET VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO ITERACTIVE FORCE VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO BOXPLOT VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO REPUTATION DEVELOPERS VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A 5 COMO A VISUALIZAÇÃO LINEAR REGRESSION VIEW APOIO A ESTIMATIVA	ESCALA DE 1 A COMO A VISUALIZAÇÃ CODE ANALYSE VIEW APOIO A ESTIMATIVA
0014454		4	4	4	4	4	4	4	5
0014301		4	4	4	4	4	5	4	5
0014080		1	1	1	1	1	1	1	1
0014378		4	4	4	4	4	5	4	5
0014312		5	4	4	4	4	5	4	5
14348		4	4	4	4	4	5	4	5
	0	4,25423728813559	3,6271186440678	3,6271186440678	3,57627118644068	3,35593220338983	4,45762711864407	3,91525423728814	4,542372881355

A seguir será detalhado cada avaliação realizada e os resultados obtidos por ela, ao final será feito uma analisa geral dos resultados de todas as medições, bem como as análises qualitativas do *framework*.

a) Medição - Sprint 3.0.15 - Desenvolvedor: DEV – 86

A Figura 47 apresenta a calibração realizada pelo DEV – 86 (os nomes foram descaracterizados por questões de confidencialidade). Após as configurações e o cálculo da estimativa de esforço de um *Sprint* (3.0.15), e os resultados finais obtidos estão representados no gráfico da Figura 48.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	45%	Tipo de Analise: Sumario
Distância de Levenshtein	20 Trocas	Anos Considerados: 99
Reputação dos Desenvolvedores	10%	Remoção de OutLiers: NAO
Observação Geral a Respeito do Experimento		
Trabalho muito bom e pratico, Todos os gráficos ficaram interessantes, Segue alguns detalhes: Code Analyser me ajudou a ver só os outliers, Reputation tava fixo, gráfico da regressão linear me confundiu com o valor que foi estimado, Box plot ajudou muito na estimativa, Diagrama de Veen não exibe intercessão entre 2 conjuntos, Fixar a configuração para todos os casos nem sempre trouxe resultado,		

Figura 47: Calibração realizada pelo participante

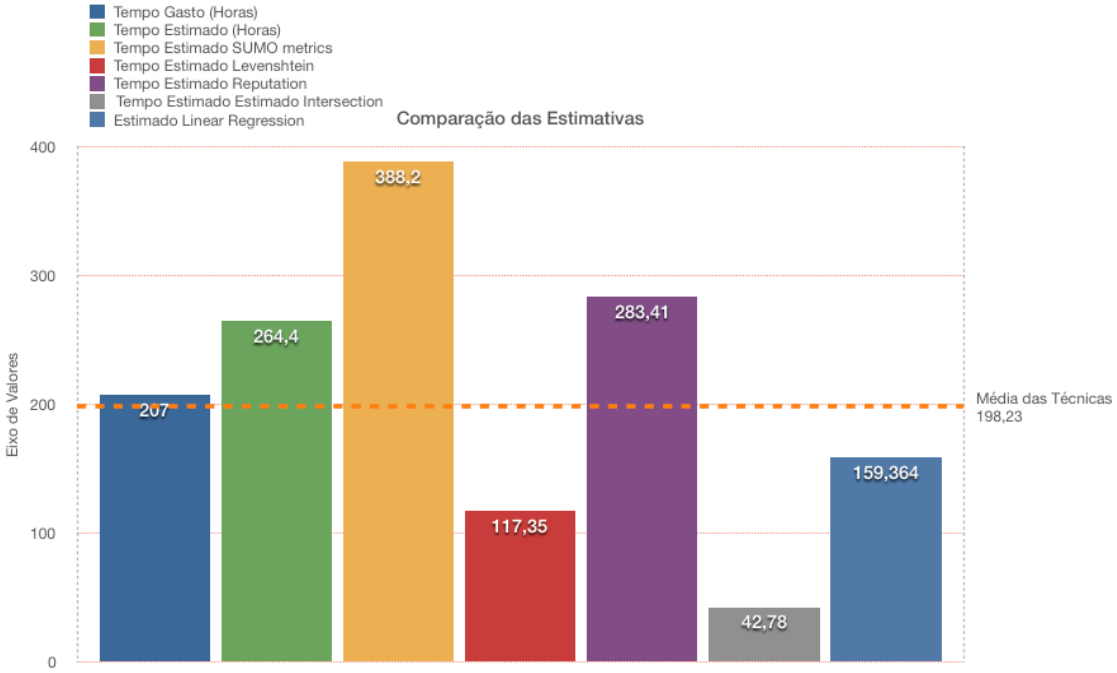


Figura 48: Resultados das estimativas realizadas pelo participante

O participante analisou também, de forma qualitativa, as técnicas e como as mesmas apoiaram a estimativa de esforço. O gráfico da Figura 49 apresenta os valores em média fornecidos pelo participante. A Figura 50 apresenta a escala de Likert (1932) utilizada para a mensuração dos valores.

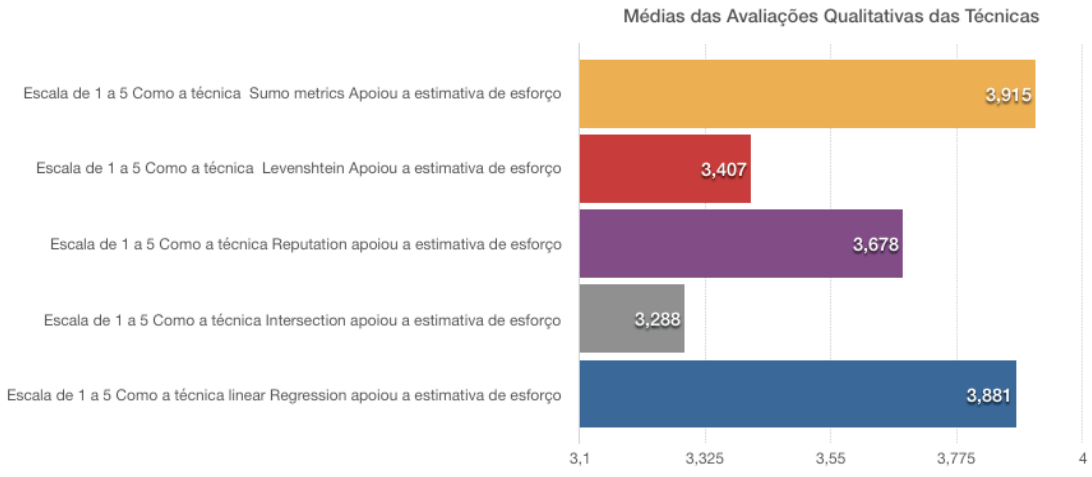


Figura 49: Resultados da análise qualitativa das técnicas realizada pelo participante

Legenda da Escala	
1	Não Apoia a Estimativa de Esforço
2	Pouco Apoio a Estimativa de Esforço
3	Indiferente
4	Apoia a Estimativa de Esforço
5	Muito Apoio a Estimativa de Esforço

Figura 50: Escala utiliza para mensuração qualitativa do experimento

O participante analisou também, de forma qualitativa, as visualizações existentes no *framework* e como as mesmas apoiaram a Estimativa de Esforço. O gráfico da Figura 51 apresenta os valores em média fornecidos pelo participante.

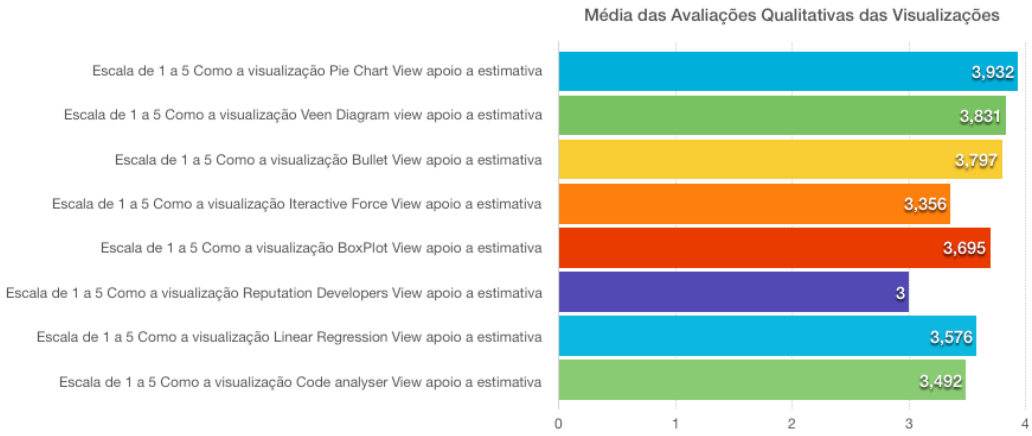


Figura 51: Resultados da análise qualitativa das Visualizações pelo participante

b) Medição - Sprint 3.0.15 - Desenvolvedor: DEV – 20

A mesma metodologia foi adotada para o participante DEV – 20. A Figura 52 apresenta a calibração assumida pelo participante e os resultados obtidos encontram-se disponíveis nas Figuras 53, 54 e 55.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	22%	Tipo de Analise: Sumario
Distância de Levenshtein	29 Trocas	Anos Considerados: 1
Reputação dos Desenvolvedores	23%	Remoção de OutLiers: SIM
Observação Geral a Respeito do Experimento		
Ao selecionar o caso, seria interessante exibir as horas gastas pelo desenvolvedor, para tomar como medida de comparação, A ferramenta tem um cunho científico muito interessante, sendo possível a disponibilidade da mesma em licença GNU (ou outra compatível), contribuiria para o meio acadêmico e permitiria empresas e grupos aperfeiçoarem a mesma,		

Figura 52: Calibração realizada pelo participante

A figura 53 apresentou valores estimados pela ferramenta em vista da calibração do desenvolvedor Dev – 20. Pode-se notar que a Técnica Sumo se mostrou mais perto dos valores gasto e estimado, apresentando aproximadamente 25,33% a menos que o valor gasto pela equipe de desenvolvimento e cerca de 60,00% menor que o estimado. Já a técnica *Linear Regression* se mostrou aproximadamente 43,76% maior que o tempo estimado e 83,62% maior que o tempo gasto.

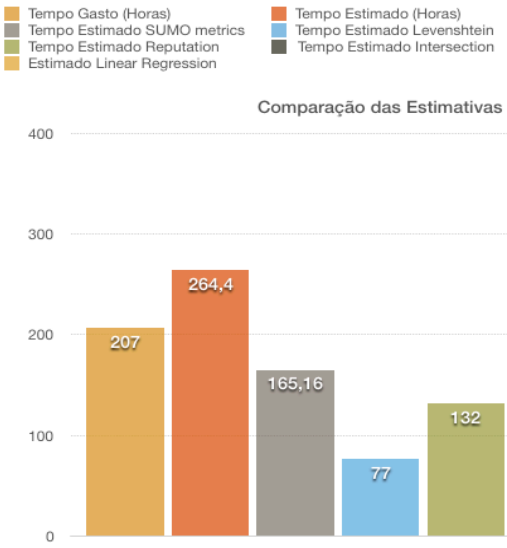


Figura 53: Resultados da análise qualitativa das técnicas realizada pelo participante

A figura 54 destaca as avaliações qualitativas das técnicas feitas pelo desenvolvedor Dev – 20. Pode-se notar que a técnica Sumo Metrics recebeu a maior avaliação média, seguido da técnica *Reputation*, ambas com uma nota maior que 4. Todas as outras receberam notas acima de 3 indicando que, para o usuário, as técnicas apresentadas na ferramenta podem apoiar a estimativa de esforço.

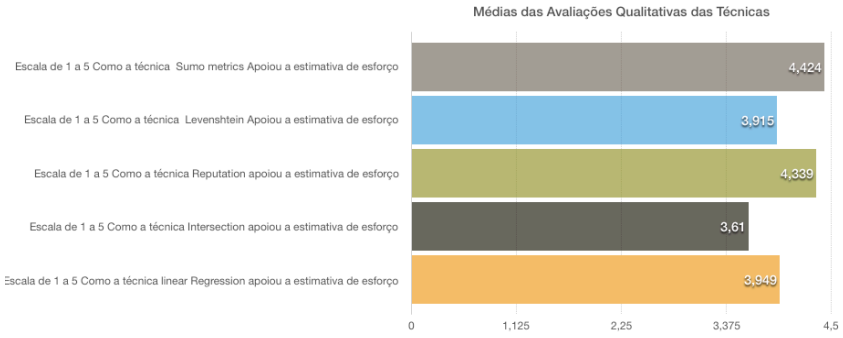


Figura 54: Resultados da análise qualitativa das Técnicas pelo participante

A figura 55 destaca as avaliações qualitativas das visualizações e como as mesmas apoiaram a estimativa de Esforço e a compreensão do software, quando o desenvolvedor Dev – 20 utilizou o *framework*. Pode-se notar que, a percepção do usuário, é de que as visualizações *Pie Chart*, *Veen Diagram* e *Code Analyser* apoiaram a estimativa de esforço e a compreensão do software. A visualização *Boxplot* foi indiferente para o usuário, juntamente com a *Bullet View* que recebeu uma nota perto de 3. As demais visualizações receberam notas acima de 4 indicando que apoiam a estimativa de esforço e a compreensão do software.

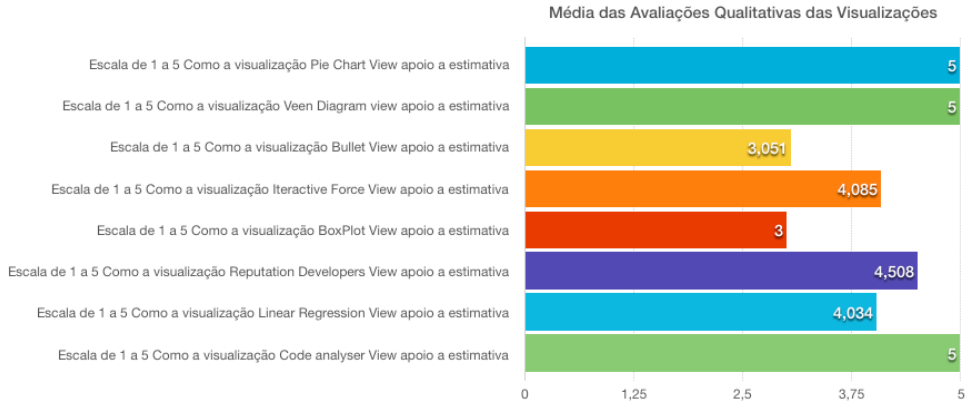


Figura 55: Resultados da análise qualitativa das visualizações pelo participante

c) Medição - Sprint 3.0.15 - Desenvolvedor: DEV – 105

Similarmente ao participante anterior, após a realização do experimento com o participante, utilizando a calibração disposta na figura 56, os resultados obtidos e suas análises, encontram-se disponíveis nas Figuras 57 a 59.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	26%	Tipo de Analise: Sumario
Distância de Levenshtein	29 trocas	Anos Considerados: 6
Reputação dos Desenvolvedores	30%	Remoção de OutLiers: SIM
Observação Geral a Respeito do Experimento		
Os graficos coloquei a maioria 3 porque não utilizei para apoiar a estimativa Sugestões - Separar as estimativas de esforço para implementação e correção de erros,Dependendo da metrica fica muito diferente, Permitir rodar varias estimativas de uma vez, Vai auxiliar a definir os parâmetros iniciais, Os gráficos seria bom serem utilizados no momento de "atribuir" a realização da tarefa , "similaridade" , "reputação" , "code analizer View",		

Figura 56: Calibração realizada pelo participante

A figura 57 demonstra a estimativa realizada pelo desenvolvedor Dev – 105. É digno de nota que com a calibração feita pelo desenvolvedor no *framework,* pôde-se obter na técnica Sumo um valor estimado 3,86% a mais que o valor gasto e 22,97% a menos que o valor estimado pela equipe, sendo que o quanto mais perto do valor real gasto melhor a métrica do *framework*. Já a técnica *Linear Regression* obteve 8,69% a mais que o valor gasto e cerca de 17,50% menor que estimado pela equipe de desenvolvimento.

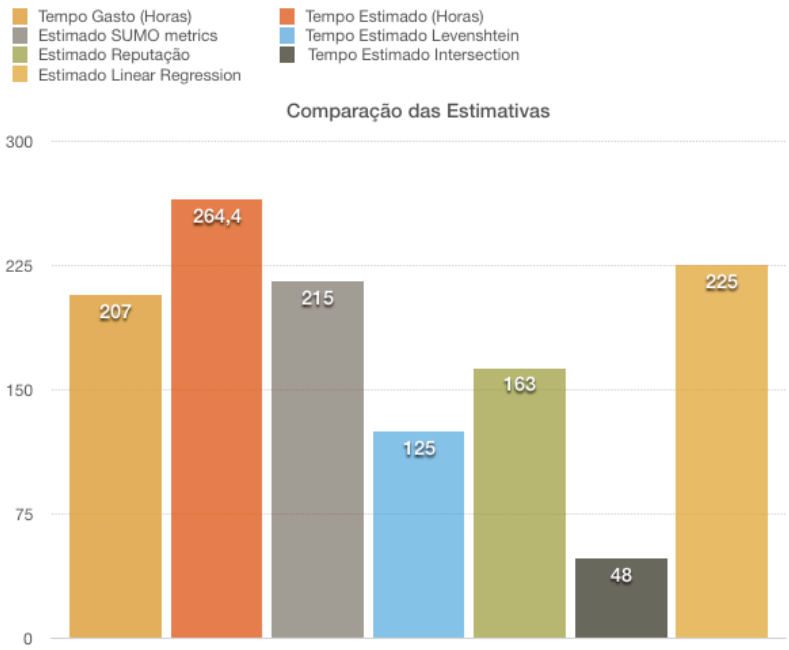


Figura 57: Resultados da análise qualitativa das técnicas realizada pelo participante

A figura 58, apresenta as avaliações qualitativas das técnicas feita pelo desenvolvedor Dev – 105. A técnica *Liner Regresion*, recebeu uma nota abaixo de 3, na média, indicando que segundo o desenvolvedor foi a técnica que menos apoiou a estimativa de esforço, mas, no entanto, conforme a análise anterior das técnicas, foi a que chegou mais perto das medições manuais feitas pela equipe de desenvolvimento. A técnica *Reputation* ficou muito perto da zona de indiferença do ponto de vista do usuário. Todas as demais técnicas receberam notas acima de 3, indicando que para o usuário as técnicas apresentadas na ferramenta apoiam de alguma forma a estimativa de esforço.

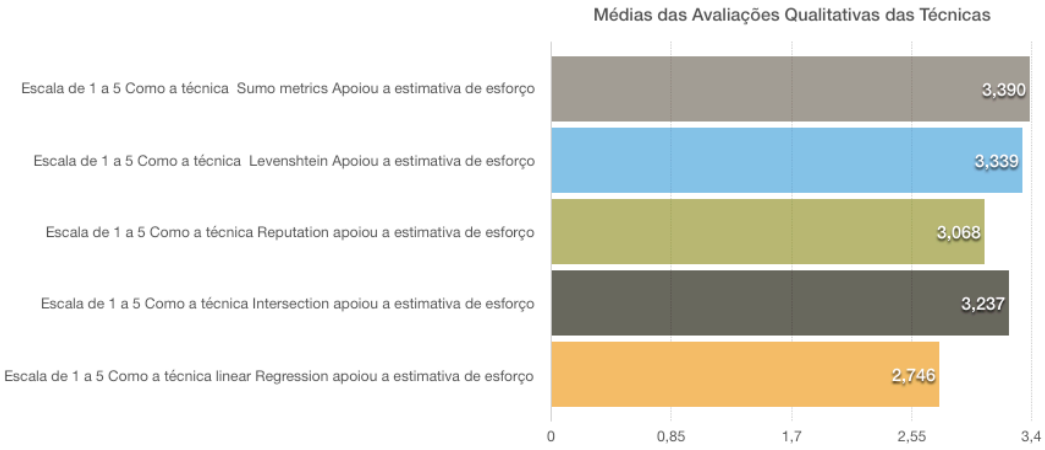


Figura 58: Resultados da análise qualitativa das Visualizações pelo participante

A figura 59 apresenta as avaliações qualitativas do Desenvolvedor 105 para as visualizações do *framework* e como as mesmas apoiaram a compreensão do software em desenvolvimento pela equipe. Pode-se notar que, na percepção do usuário, as visualizações *Interactive Force* e *Code Analyser* apoiaram a estimativa de esforço e a compreensão do software. A visualização *Pie Chart* e *Veen Diagram* ofereceram, segundo o usuário, pouco apoio à estimativa de esforço e compreensão. Já a visualização *Reputation* entrou na zona de indiferença para o usuário. As demais receberam notas acima de 4, indicando que apoiam a estimativa de esforço e a compreensão do software.

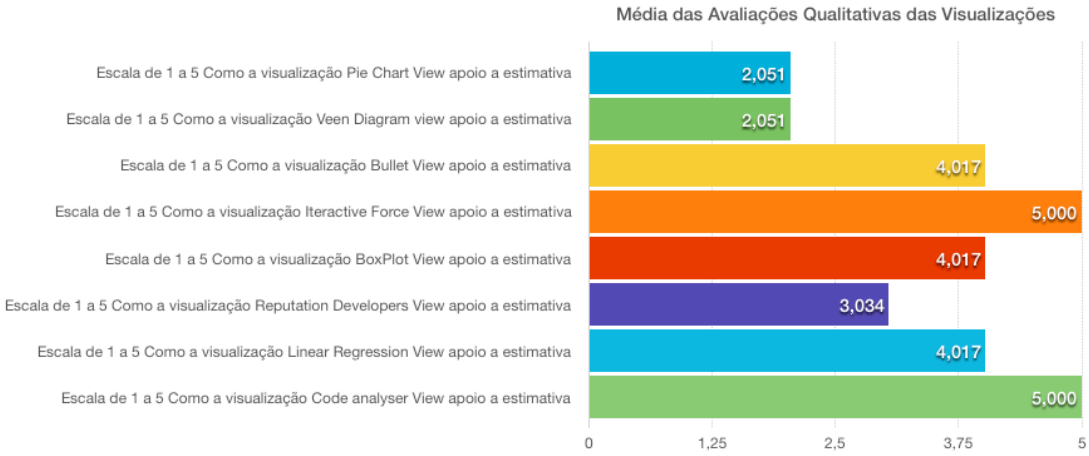


Figura 59: Resultados da análise qualitativa das visualizações pelo participante

d) Medição - Sprint 3.0.15 - Desenvolvedor: DEV – 95

Similarmente ao participante anterior, após a execução do experimento com o participante DEV – 95, os resultados obtidos pela calibração realizada e apresentada na Figura 60, encontram-se disponíveis junto com suas analises nas figuras 61 a 63.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	30%	Tipo de Analise: Sumario
Distância de Levenshtein	29 Trocas	Anos Considerados: 2
Reputação dos Desenvolvedores	30%	Remoção de OutLiers: SIM
Observação Geral a Respeito do Experimento		

Figura 60: Calibração realizada pelo participante

Conforme apresentado na figura 61 o desenvolvedor Dev – 95, obteve com a técnica Sumo Metrics um valor aproximado de 3,68% a menos que o valor estimado pelos desenvolvedores e 23,18% a menos que o valor gasto pela equipe. As demais técnicas ficaram de uma forma geral 73,00% e 200% respectivamente menores que o estimado e gasto pela equipe de desenvolvimento. Essas diferenças se devem às calibrações realizadas pelo usuário em cada uma das técnicas o que condicionou sua avaliação à obtenção de tais valores.

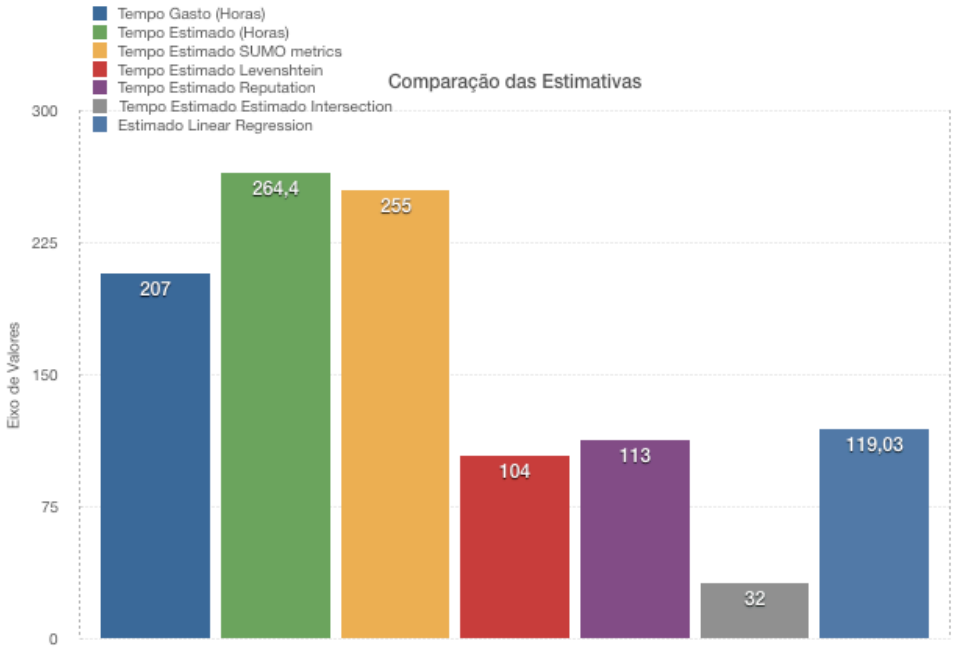


Figura 61: Resultados das estimativas realizadas pelo participante

A figura 62 destaca as avaliações qualitativas das técnicas feitas pelo desenvolvedor Dev – 95. Pode-se notar que a técnica Sumo Metrics recebeu a maior avaliação média 4,203. A técnica *Intersection* ficou próxima da zona de indiferença. A técnica *Linear Regression*, segundo a opinião do usuário, apoiou pouco a estimativa de esforço e compreensão do software e as demais receberam notas acima de 3,5, indicando que para o usuário as técnicas apresentadas na ferramenta apoiaram a estimativa de esforço.

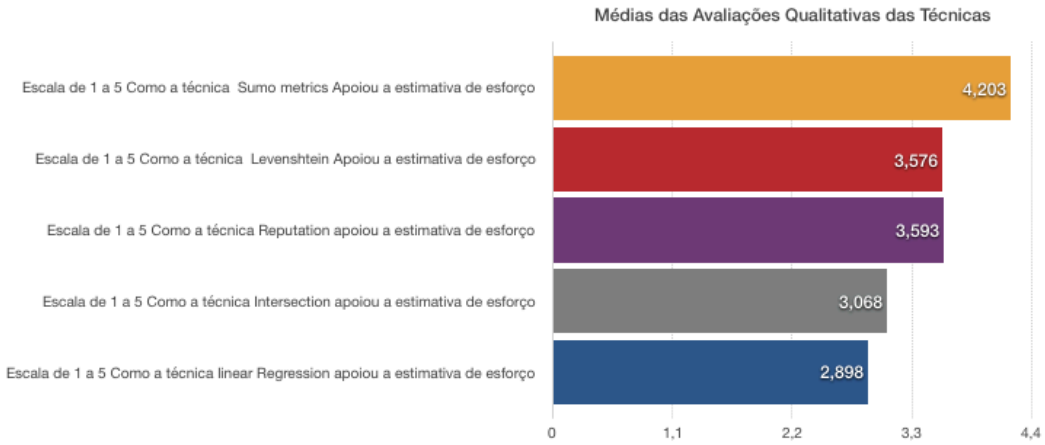


Figura 62: Resultados da análise qualitativa das técnicas realizada pelo participante

A figura 63 apresenta as análises qualitativas das visualizações e como as mesmas apoiaram a estimativa de esforço e a compreensão do software, quando o desenvolvedor Dev – 95 utilizou o *framework*. Pode-se notar que, na percepção do usuário, as visualizações *Pie Chart*, *Reputation Developers* e *Code Analyser* apoiaram a estimativa de esforço e a compreensão do software, pois receberam notas acima de 4. A visualização *Boxplot* ficou próxima da área de indiferença. E todas as demais receberam notas acima de 3,5 indicando que apoiaram a estimativa de esforço e a compreensão do software.

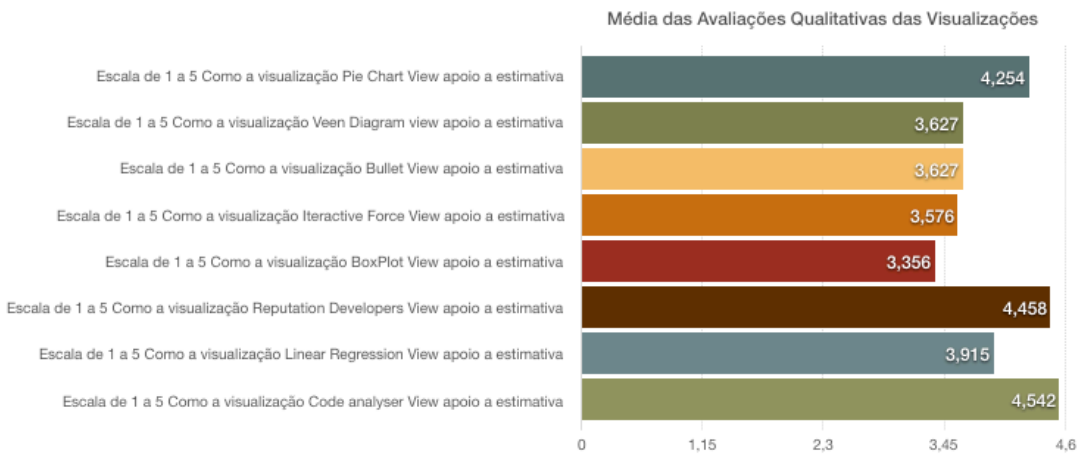


Figura 63: Resultados da análise qualitativa das Visualizações pelo participante

e) Medição - Sprint 3.0.15 - Desenvolvedor: DEV – 94

Após a condução do experimento com o participante DEV – 94, A figura 64 apresenta os dados da calibração realizada pelo usuário que levou os resultados apresentados e a analisados nas figuras 65 a 67.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	50%	Tipo de Analise: Sumario
Distância de Levenshtein	56 Trocas	Anos Considerados: 99
Reputação dos Desenvolvedores	40%	Remoção de OutLiers: NÃO
Observação Geral a Respeito do Experimento		

Figura 64: Calibração realizada pelo participante

As medições encontradas pelo desenvolvedor Dev – 94 são apresentadas na figura 65. Os destaques dessa análise ficam para as técnicas Levenshtein e Reputation. A primeira obteve um valor aproximado de 10,21% a maior que o valor Estimado pelos desenvolvedores e 40,77% a maior que o valor Gasto pela equipe. A técnica Levenshtein apresentou 16,64% a mais que o valor estimado pelos desenvolvedores e 48,98% a maior que o gasto pela equipe. O valor maior que 200%, do que o valor estimado, encontrado pela técnica Levenshtein é justificado pelo número de trocas assumido pelo desenvolver em sua calibração, 56 trocas, o que diminui o grau de similaridades entre os casos, encontrando mais resultados e consequentemente maior tempo de estimativa. Outro destaque é que o usuário não utilizou a remoção de *outliers*

o que permite encontrar requisições de mudanças que estão muito acima da média. As demais técnicas ficaram abaixo de 32,00% do real gasto e 66,00% a menor que o Estimado pela equipe de desenvolvimento.

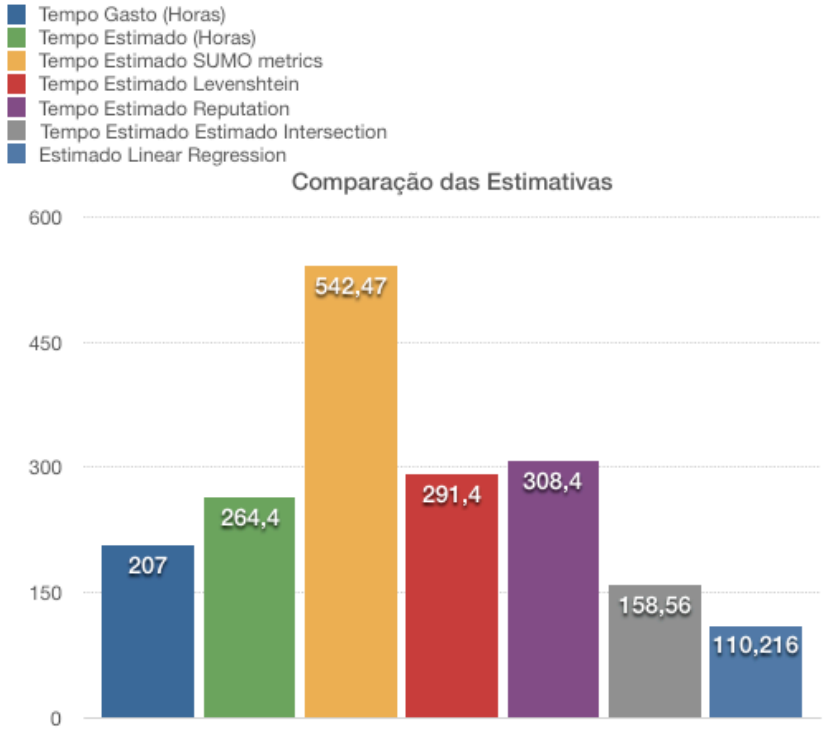


Figura 65: Resultados das estimativas realizadas pelo participante

A figura 66 apresenta as avaliações qualitativas das técnicas feita pelo desenvolvedor Dev – 94. Segundo a percepção do usuário apenas a técnica Sumo Metrics, recebeu uma nota 4, indicando que a mesma apoia a estimativa de esforço. Todas as demais técnicas ficaram perto da zona de indiferença do ponto de vista do usuário. Mas conforme a análise anterior das técnicas *Reputation* e *Levenshtein*, as mesmas se aproximaram muito do real gasto e estimado. Uma linha de evidência para essa análise qualitativa por parte do usuário, diferente do encontrado do valor esperado, se deve ao fato de que ele juntamente com os demais participantes, não receberam os valores estimados e gasto pela equipe para que comparassem com seus resultados ou calibrassem a ferramenta. Isso foi feito de forma proposital, para que se pudesse capitar a percepção do desenvolvedor e a sua experiência sobre as estimativas a serem realizadas.

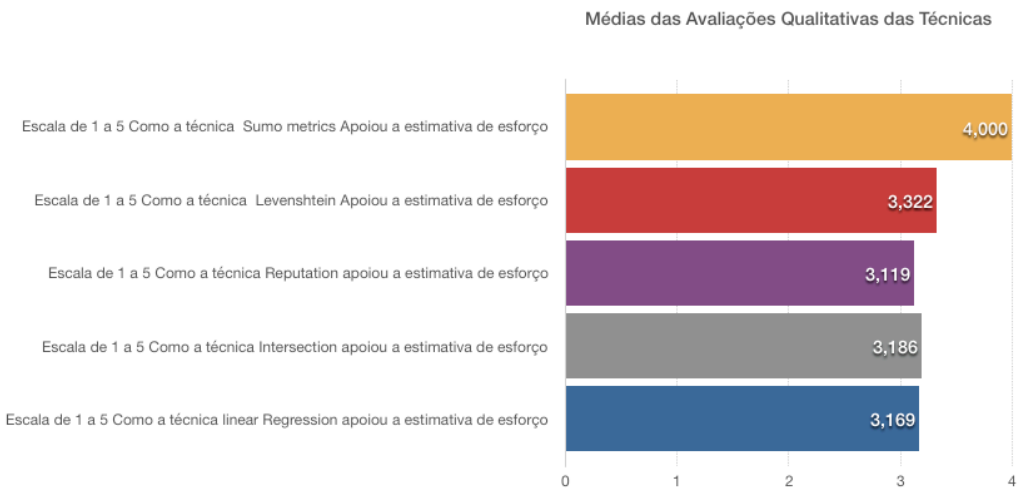


Figura 66: Resultados da análise qualitativa das técnicas realizada pelo participante

A figura 67 apresenta as avaliações qualitativas do Desenvolvedor 94 para as visualizações do *framework* e como as mesmas apoiaram a compreensão do software pelo participante. Pode-se notar que na percepção do usuário, as visualizações *Pie Chart*, *Veen Diagram* e *Bullet View* estão na zona de indiferença quanto ao apoio à visualização e compreensão do software. As demais técnicas estão acima de 3, evidenciando que, segundo a percepção do participante, apoiam em média a estimativa de esforço.

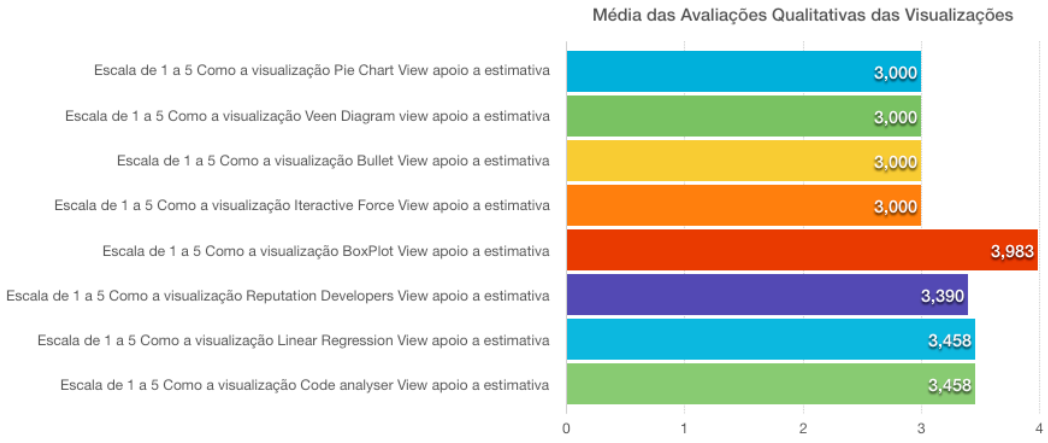


Figura 67: Resultados da análise qualitativa das visualizações pelo participante

f) Medição - Sprint 3.0.15 - Desenvolvedor: GER – 1

A figura 68 apresenta a calibração realizada pelo Gerente de Projetos. Vale ressaltar que o mesmo não possuía qualquer conhecimento a respeito da base de dados ou das técnicas aplicadas pela empresa. Após as devidas configurações e o cálculo da Estimativa de Esforço de um Sprint (3.0.15), os resultados finais obtidos estão representados nas figuras 69 a 71.

Configuração da Ferramenta		MEDIÇÃO 1
Percentual Similaridade	17%	Tipo de Analise: Sumario
Distância de Levenshtein	98 Trocas	Anos Considerados: 2
Reputação dos Desenvolvedores	22%	Remoção de OutLiers: SIM
Observação Geral a Respeito do Experimento		
Em várias análises a Mediamna exibida no boxsplot apresentou -se mais precisa que a médiaa exibida nos resultados,, talvez seja interessante exibir também esse número. - Os parâmetroos que calibram mas demais análises não funcionaram para a regressão linear, talvez pelo período utilizado de 1 ou 2 anos anteriores - obtive muito poucos resultados com o sumário sendo analisado, talvez porque apresentam mais variações que os sumários - fiquei surpreso com a assertividade da análise por reputação.		

Figura 68: Calibração e observações realizadas pelo participante

A figura 69 demonstra a estimativa realizada pelo Gerente de Projetos Ger – 1. Pode-se notar que a técnica *Reputation* apresentou valor aproximado de 15,73% a maior que o valor estimado e 47,82% a maior que o valor gasto pela equipe. A técnica Levenshtein apresentou um valor 18,96% menor que o tempo gasto e 51,95% a menor que o estimado pela equipe de desenvolvimento. A técnica Sumo apresentou 51,66% a mais que o estimado e 93,71% a mais que o gasto. Isso pode ser justificado devido à redução do percentual de similaridade configurado pelo usuário. As demais técnicas obtiveram valores abaixo de 50% do tempo gasto e 90% do estimado.

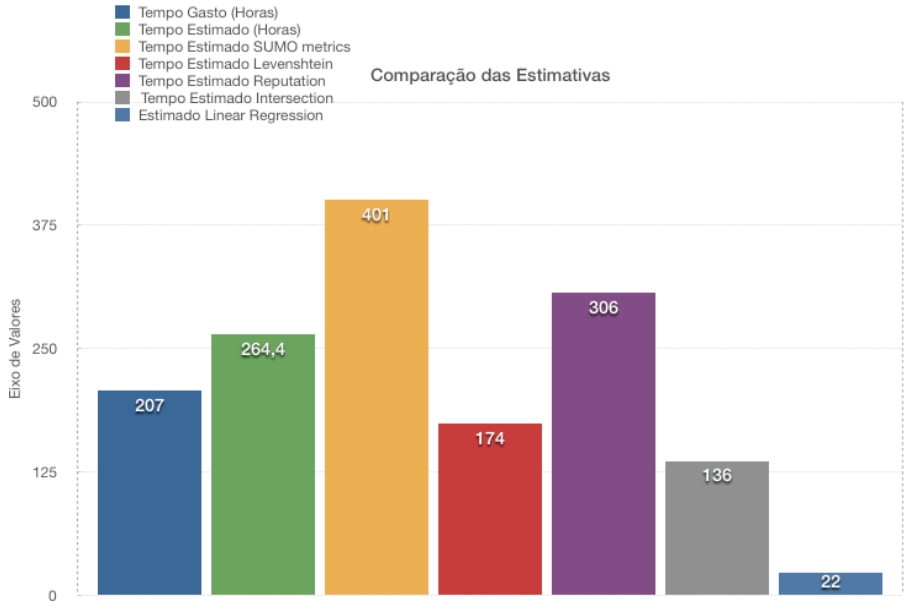


Figura 69: Resultados das estimativas realizadas pelo participante

A figura 70 apresenta as avaliações qualitativas das técnicas feita pelo Gerente de Projetos Ger – 1. A técnica Liner Regression, recebeu uma nota abaixo de 3 na média. Segundo o gerente de projetos, foi a técnica que menos apoiou a estimativa de esforço. Todas as demais técnicas receberam notas acima de 3,9 indicando que, segundo a percepção do Gerente de Projetos, todas elas apoiam de alguma forma a estimativa de esforço.

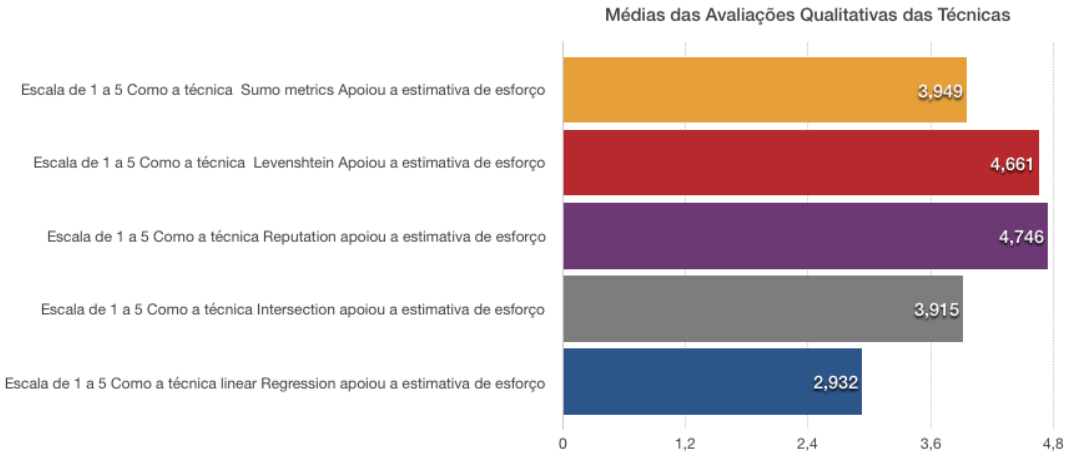


Figura 70: Resultados da análise qualitativa das técnicas realizada pelo participante

A figura 71 apresenta as análises qualitativas das visualizações e como as mesmas apoiaram a estimativa de Esforço e a compreensão do software quando o Gerente de Projeto - Ger. – 1 - utilizou o *framework*. Pode-se notar que, na percepção do usuário, as visualizações *Pie Chart*, *Veen Diagram* ficaram próximas da área de indiferença no apoio a estimativa de esforço e da compreensão do software. Todas as demais visualizações receberam notas acima de 3,9 indicando que, segundo Gerente de Projeto, apoiam a estimativa de esforço e a compreensão do software. Vale notar que algumas Visualizações como *Boxplot*, *Reputation Developers* e *Code Analyser*, receberam nota máxima, ou seja, segundo a percepção do usuário, provem muito apoio a estimativa de esforço e a compreensão do software em desenvolvimento.

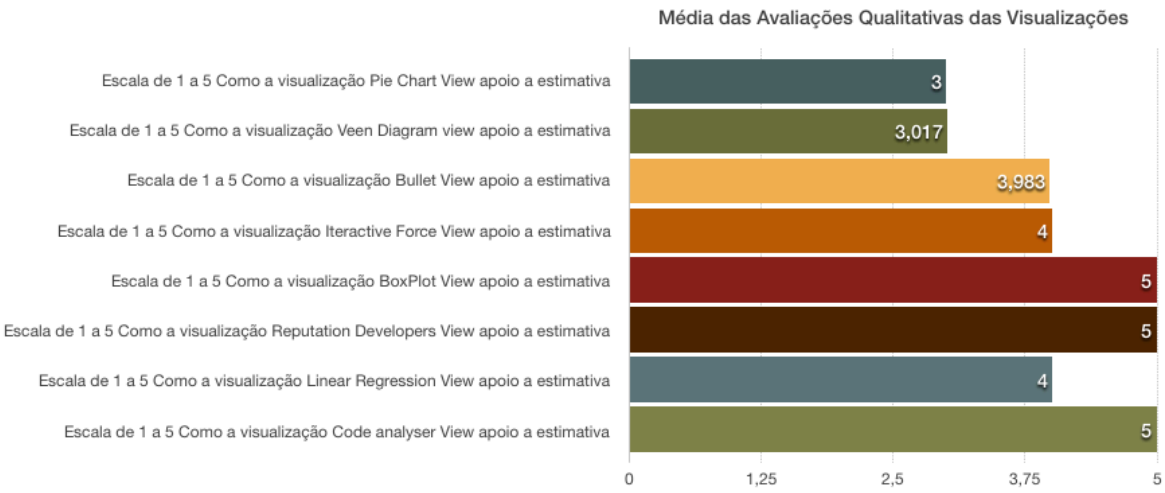


Figura 71: Resultados da análise qualitativa das Visualizações pelo participante

g) Resultados Gerais

A Tabela 14 apresenta a comparação das técnicas em função das medições de cada participante, bem como a média das técnicas, após a condução do experimento.

Tabela 14: Comparação das técnicas em função das medições

VALORES FIXOS	ESTIMADO PMG - EQUIPE		190,02			
	TEMPO GASTO - EQUIPE		264,40			
DESCRIÇÃO	DEV - 95	DEV - 86	DEV - 20	DEV - 105	DEV - 94	GER - 1
Sumo Metrics	255,00	388,20	165,16	215,00	542,47	401,00
Levenshtein	104,00	117,35	77,00	125,00	291,40	174,00
Reputation	113,00	283,41	132,00	163,00	308,40	306,00
Intersection	32,00	42,78	66,00	48,00	158,56	136,00
Linear Regression	119,03	159,36	380,11	225,00	110,22	22,00
MÉDIA DAS TÉCNICAS	124,61	198,22	164,05	155,20	282,21	207,80

A Figura 72 apresenta de forma gráfica a comparação das técnicas em função das medições dos participantes. Pode-se observar na mesma, a convergência de valores em determinadas técnicas, bem como, a discrepância de algumas medições realizadas pelos participantes, indicando que calibrações diferentes e a compreensão das técnicas podem gerar valores de estimativas diversos. Isso reforça o fato da necessidade de um especialista que possa calibrar a ferramenta de uma forma que atenda o perfil da empresa e às necessidades de sua equipe.

Em relação ao tempo gasto, a Figura 72 demonstra que na técnica Sumo Metrics, 4 participantes estiveram próximos, ou acima do valor real, e 2 deles abaixo. A técnica Levenshtein apresentou 5 participantes abaixo do valor gasto e 1 acima desse valor. Já a técnica *Reputation*, obteve 3 participantes acima do gasto e 3 participantes abaixo desse valor. Todos os participantes obtiveram na técnica *Intersection* valores abaixo do real gasto. Já a técnica *Linear Regression* obteve 1 valor acima do Gasto e 5 participantes abaixo do mesmo.

Na média geral das técnicas apenas um participante esteve acima do Valor Gasto todos os demais estiveram abaixo conforme pode-se observar. Pode-se destacar, na análise análoga ao tempo estimado, que 3 participantes estiveram bem próximos ou acima do valor estimado e 3 participantes abaixo do mesmo na média geral. Além disso, na técnica Sumo Metrics 5 participantes estiveram acima do valor Estimado e apenas 1 abaixo do mesmo.

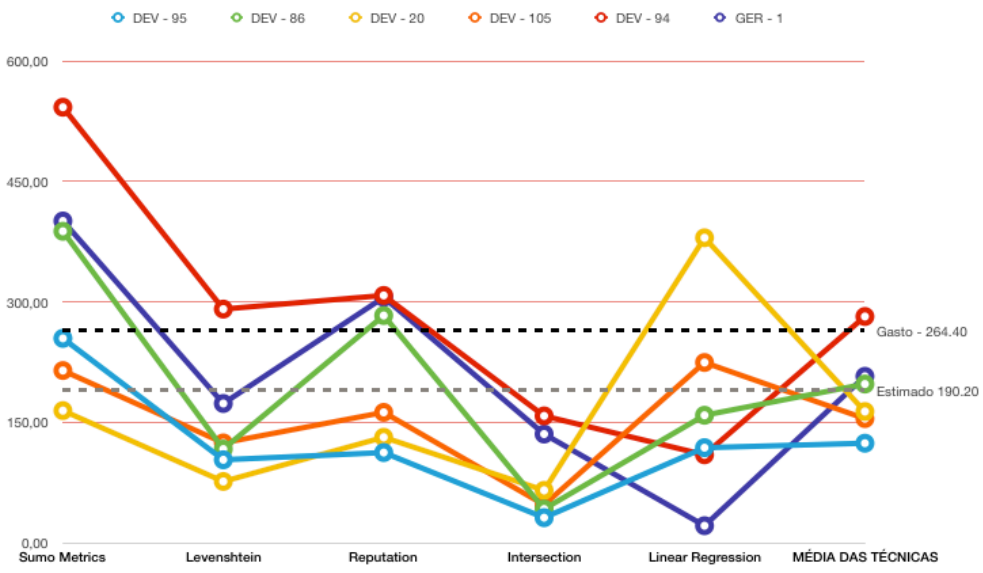


Figura 72: Comparativo gráfico de participantes em função das técnicas

A Figura 73 permite um comparativo entre as técnicas por participante. Podendo-se observar as divergências entre valores encontrados pelos participantes em função das calibrações efetuadas pelo *framework*. Pode-se notar que o participante Dev – 94 obteve os valores mais aproximados ao tempo gasto, excetuando-se a técnica Sumo Metrics, que foi a mais discrepante entre os participantes.

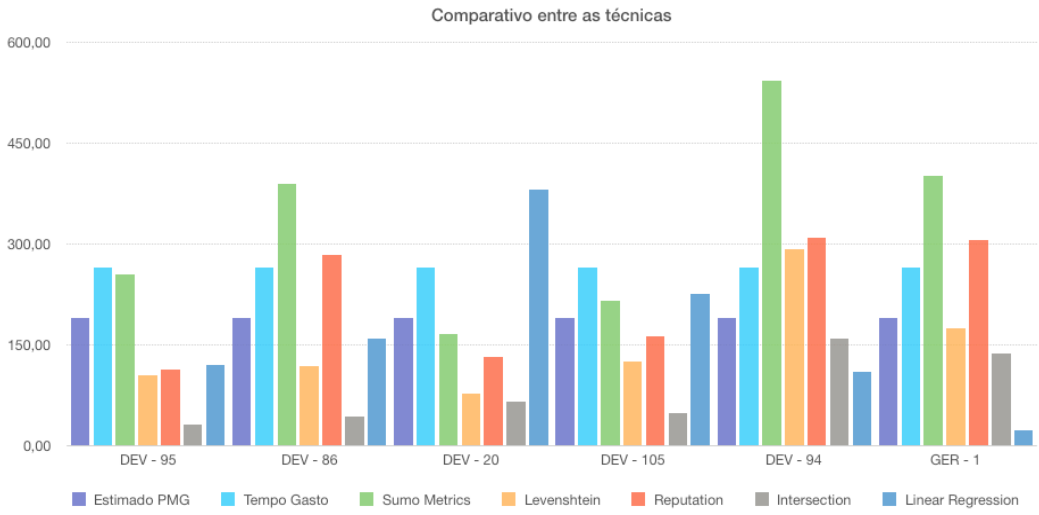


Figura 73: Comparativo gráfico entre as técnicas por participante

Mediante os dados coletados (Tabela 12), foram realizadas as análises estatísticas, e seus resultados foram utilizados para verificar a existência de diferenças significativas entre os valores Previsto, Realizado e as Técnicas de Estimativa de Esforço utilizadas pelo *framework*. Diante disso, foram definidas as seguintes hipóteses para análise estatística:

H0: **Não existe** diferença entre as médias de Estimativas de Tempo Previstas e Realizadas e as técnicas Sumo, Levenshtein, Reputation, Intersection e Linear Regression.

H1: **Existe** diferença entre as médias de Estimativas de Tempo Previstas e Realizadas e as técnicas Sumo, Levenshtein, Reputation, Intersection e Linear Regression.

Uma análise estatística inicial foi a geração do *Boxplot* das técnicas (Sumo, Levenshtein, Reputation, Intersection e Linear Regression), conforme exibido na Figura 74. O *Boxplot* (Sirqueira et al., 2016) é um gráfico utilizado para avaliar a distribuição empírica dos dados, sendo formado pelo primeiro e terceiro quartil, e pela mediana. Analisando o *Boxplot*, pode-se observar que as médias das técnicas se encontram muito próximas das linhas Tempo Gasto (Vermelho) e Tempo Estimado (Verde), evidenciando uma proximidade das técnicas apresentadas e os valores Estimados e Gastos pela equipe de desenvolvimento.

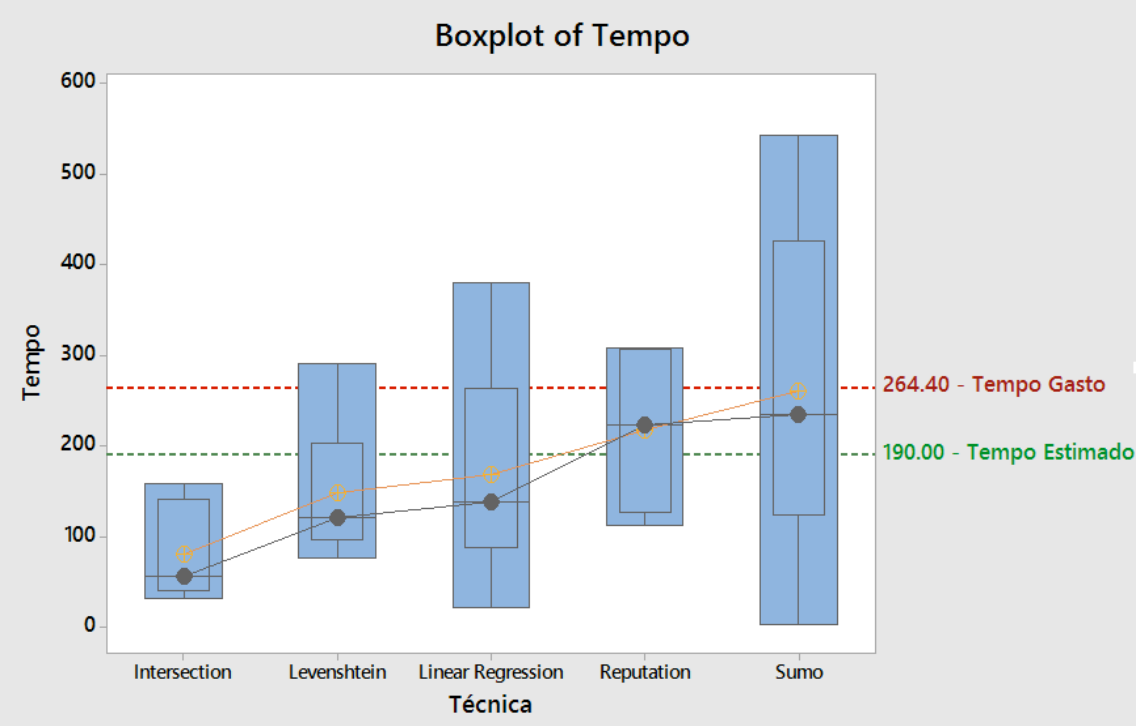


Figura 74: *Boxplot* das técnicas em função do Tempo Gasto e Estimado PMG

Foram analisadas, ainda a partir da Tabela 14, as colunas Tempo (total de tempo encontrado em cada uma das técnicas) em relação às Técnicas (Tempo Estimado PMG, Tempo Gasto, Sumo, Levenshtein, Reputation, Intersection e Linear Regression) apresentadas pelo *framework* e, em seguida, foram aplicados testes estatísticos nos dados coletados, sendo utilizado um nível de significância de 5% ($\alpha = 0{,}05$). A significância estatística de um resultado é uma medida estimada do grau de veracidade do resultado, ou seja, o valor do nível-p (*p-Value)* representa um índice decrescente da confiabilidade de um resultado (Sirqueira et al., 2016).

Para essa análise, foi utilizado um teste paramétrico para comparação das médias, entretanto, para isso, deve-se verificar a normalidade e homocedasticidade dos dados. Para o teste de normalidade foi usado o teste estatístico de *Shapiro-Wilk*, visto a quantidade de amostras serem inferior a 50 elementos. Assumindo-se um nível de significância de 5%, foi encontrado um *p-value* 0.100 (Figura 75), indicando que a distribuição dos dados é normal, portanto, foi utilizado o teste homocedasticidade de *Levene* (Sirqueira et al., 2016).

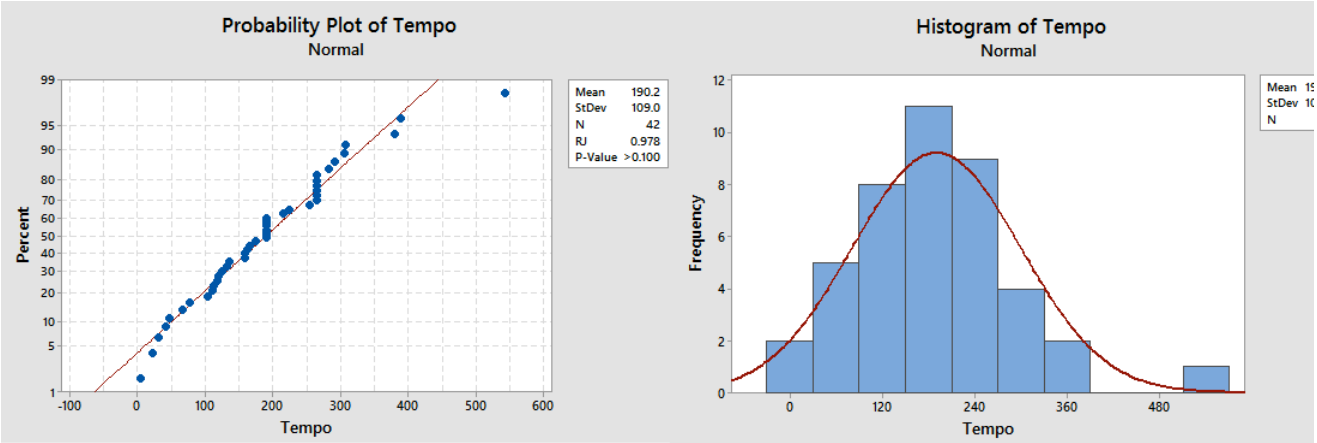


Figura 75: Teste de normalidade sobre a variável Tempo.

Considerando-se esse teste para Igualdade de Variâncias, pode-se observar que as amostras são homocedásticas (Figura 76), assumindo que o *p-value* apresentado foi menor que 5%, o que permitiu a utilização de um teste paramétrico para comparação das médias.

Test and CI for One Variance: Tempo

Method

Null hypothesis σ = 0.05
Alternative hypothesis σ > 0.05

The chi-square method is only for the normal distribution.
The Bonett method is for any continuous distribution.

Statistics

Variable	N	StDev	Variance
Tempo	42	109	11884

95% One-Sided Confidence Intervals

Variable	Method	Lower Bound for StDev	Lower Bound for Variance
Tempo	Chi-Square	93	8557
	Bonett	87	7583

Tests

Variable	Method	Test Statistic	DF	P-Value
Tempo	Chi-Square	1.94899E+08	41	0.000
	Bonett	—	—	0.000

Figura 76: Teste de igualdade de variância sobre a variável Tempo

Como teste paramétrico foi utilizado o teste ANOVA (Sirqueira et al., 2016), visto que, para a análise da amostra temos um fator (Tempo) e mais de dois tratamentos (técnicas). Mediante a aplicação do respectivo teste, a um nível de significância de 5%, nota-se, na Figura 77, que, com um *p-value* igual a 0,030, pode-se aceitar a hipótese nula de que as médias são estatisticamente iguais. Mediante a realização do referido teste, ainda é possível obter uma análise gráfica do teste ANOVA, com a comparação das médias em função do tratamento, conforme demonstrado na Figura 78.

One-way ANOVA: Tempo versus Técnica

Method

Null hypothesis	All means are equal
Alternative hypothesis	At least one mean is different
Significance level	α = 0.05

Equal variances were assumed for the analysis.

Factor Information

Factor	Levels	Values
Técnica	7	Estimado PMG, Gasto, Intersection, Levenshtein, Linear Regression, Reputation, Sumo

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Técnica	6	153555	25593	2.68	0.030
Error	35	333692	9534		
Total	41	487248			

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
97.6426	31.51%	19.77%	1.38%

Means

Técnica	N	Mean	StDev	95% CI
Estimado PMG	6	190.0	0.0	(109.1, 270.9)
Gasto	6	264.4	0.0	(183.5, 345.3)
Intersection	6	80.6	53.3	(-0.4, 161.5)
Levenshtein	6	148.1	77.1	(67.2, 229.0)
Linear Regression	6	169.3	122.8	(88.4, 250.2)
Reputation	6	217.6	91.3	(136.7, 298.6)
Sumo	6	261.6	185.9	(180.7, 342.6)

Pooled StDev = 97.6426

Figura 77: Análise do teste paramétrico ANOVA

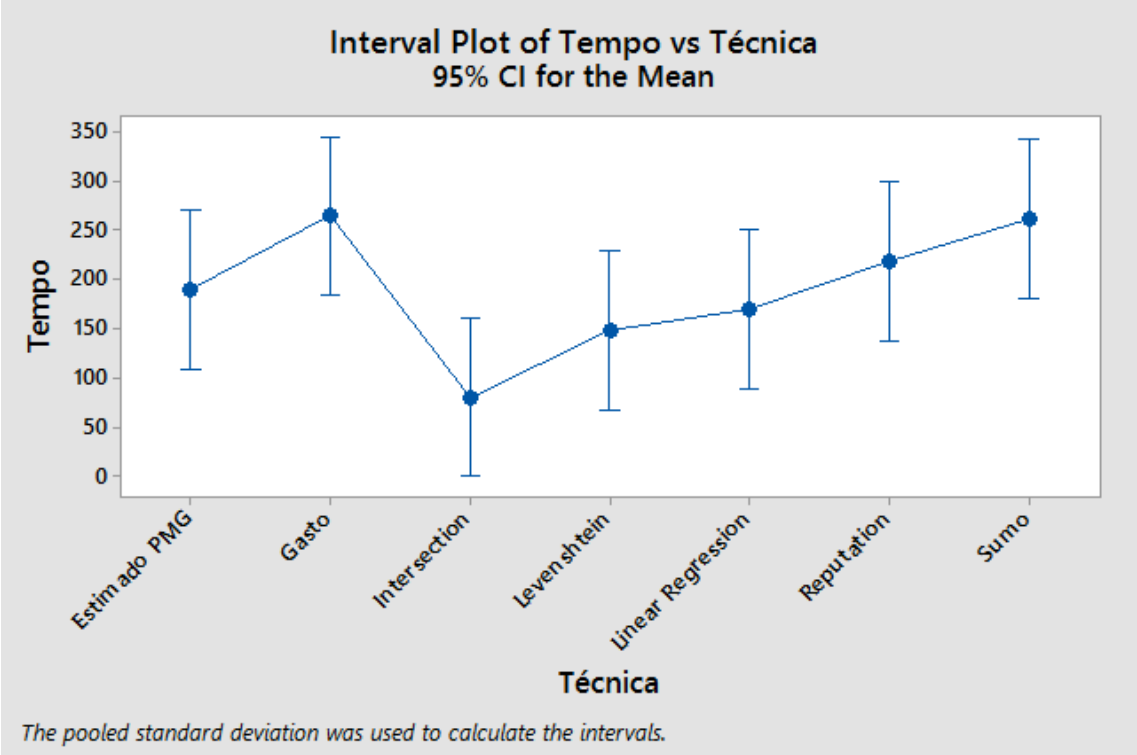


Figura 78: Análise gráfica do teste paramétrico ANOVA

Dado que, no experimento realizado os participantes ficaram livres para configurar e calibrar o *framework,* notou-se que por meio dos cálculos estatísticos conduzidos pela pesquisa não houve uma sensível diferença entre as médias. Com isso reforçamos o valor do *framework* e sua importância para o contexto de estimativa de esforço em atividades de manutenção e compreensão de software.

h) Análise Qualitativa

Além das análises quantitativas realizadas durante o estudo experimental, foram realizadas análises qualitativas. A Tabela 15 apresenta a comparação entre as análises qualitativas de cada uma das técnicas em função dos desenvolvedores que as qualificaram, bem como, a média geral atribuída a cada uma das técnicas.

Tabela 15: Comparação das análises qualitativas das técnicas

DEV/ TÉCNICA	COMO SUMO METRICS APOIOU A ESTIMATIVA DE ESFORÇO	COMO A TÉCNICA LEVENSHTEIN APOIOU A ESTIMATIVA DE ESFORÇO	COMO A TÉCNICA REPUTATION APOIOU A ESTIMATIVA DE ESFORÇO	COMO A TÉCNICA INTERSECTION APOIOU A ESTIMATIVA DE ESFORÇO	COMO A TÉCNICA LINEAR REGRESSION APOIOU A ESTIMATIVA DE ESFORÇO
DEV - 86	3,92	3,41	3,68	3,23	3,88
DEV - 20	4,42	3,92	4,34	3,61	3,95
DEV - 105	3,39	3,34	3,07	3,24	2,75
DEV - 95	4,20	3,58	3,59	3,07	2,90
DEV - 94	4,00	3,32	3,12	3,19	3,17
GER - 1	3,95	4,66	4,75	3,92	2,93
MÉDIA	3,98	3,70	3,76	3,37	3,26

O gráfico, apresentado na Figura 79, ilustra como cada desenvolvedor qualificou o apoio oferecido por cada técnica à estimativa de esforço realizada pelo *framework*. Pode-se notar que em média, todas as técnicas receberam valores superiores a 3 (três), indicando que estão acima da área de indiferença proposta pela escala de Likert (Figura 49). Portanto, de um modo geral as técnicas apresentadas pelo *framework*, na opinião dos participantes do experimento, apoiam a estimativa de esforço.

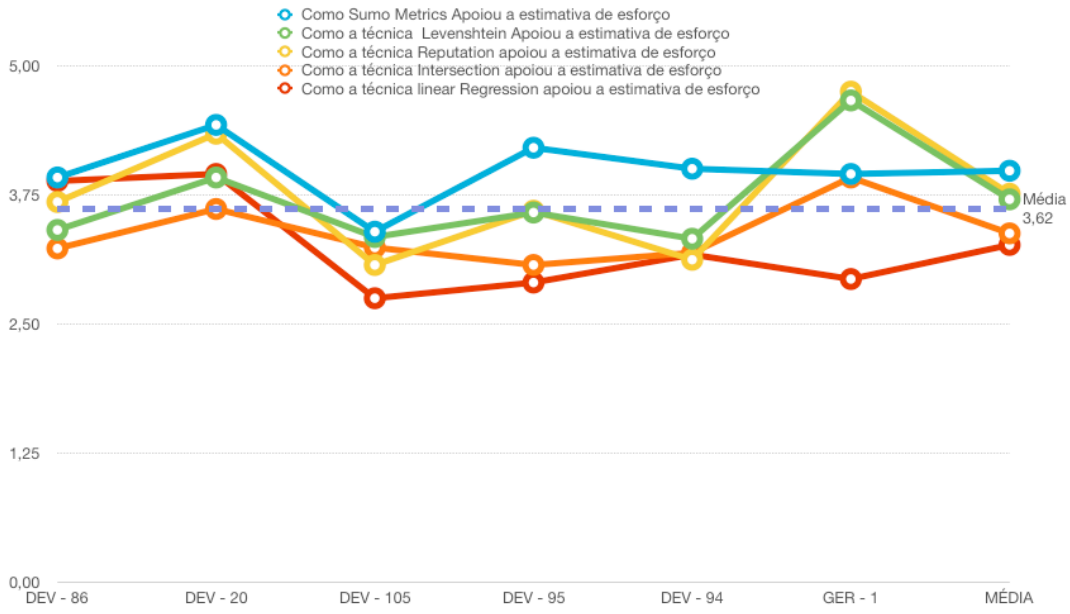


Figura 79: Gráfico de análise qualitativa das técnicas.

Uma segunda análise qualitativa também foi proposta, com o intuito de avaliar a opinião dos participantes em relação às visualizações existentes no *framework*. A Tabela 16 apresenta a média dos valores informados por cada desenvolvedor em função das visualizações apresentadas. O objetivo era descobrir quais as visualizações

apoiaram os participantes do experimento, à medida que utilizavam o *framework* para as medições de estimativa de esforço.

Tabela 16: Comparação das análises qualitativas das visualizações

DEV / VIEW	COMO A VISUALIZAÇÃO PIE CHART VIEW APOIOU A ESTIMATIVA	COMO A VISUALIZAÇÃO VEEN DIAGRAM VIEW APOIOU A ESTIMATIVA	COMO A VISUALIZAÇÃO BULLET VIEW APOIOU A ESTIMATIVA	COMO A VISUALIZAÇÃO ITERACTIVE FORCE VIEW APOIOU A ESTIMATIVA	COMO A VISUALIZAÇÃO BOXPLOT VIEW APOIOU A ESTIMATIVA	COMO A VISUALIZAÇÃO REPUTATION DEVELOPERS VIEW Apoiou A ESTIMATIVA	COMO A VISUALIZAÇÃO LINEAR REGRESSION VIEW Apoiou A ESTIMATIVA	COMO A VISUALIZ CODE ANALYSER Apoiou A ESTIMA
DEV - 86	3,93	3,83	3,80	3,36	3,70	3,00	3,58	3,49
DEV - 20	5,00	5,00	3,05	4,08	3,00	4,50	4,03	5,00
DEV - 105	2,05	2,05	4,02	5,00	4,02	3,03	4,02	5,00
DEV - 95	4,25	3,63	3,63	3,58	3,36	4,46	3,92	4,54
DEV - 94	3,00	3,00	3,00	3,00	3,98	3,39	3,46	3,46
GER - 1	3,00	3,02	3,98	4,00	5,00	5,00	4,00	5,00
MÉDIA	3,54	3,42	3,58	3,84	3,84	3,90	3,83	4,42

A Figura 80 apresenta uma análise média das visualizações. Todas as visualizações mostraram estar acima da área de indiferença, proposta pela escala de Likert (Figura 50), no entanto, algumas visualizações como *Pie Chart View, Veen Diagram View* e *Bullet View* ficaram abaixo da média geral, indicando que, para os participantes, foram menos importantes no apoio à estimativa de esforço que as demais. Portanto, mediante a análise feita, existem indícios que as visualizações apresentadas pela ferramenta, em sua maioria, apoiam a estimativa de esforço e a compreensão do software em desenvolvimento.

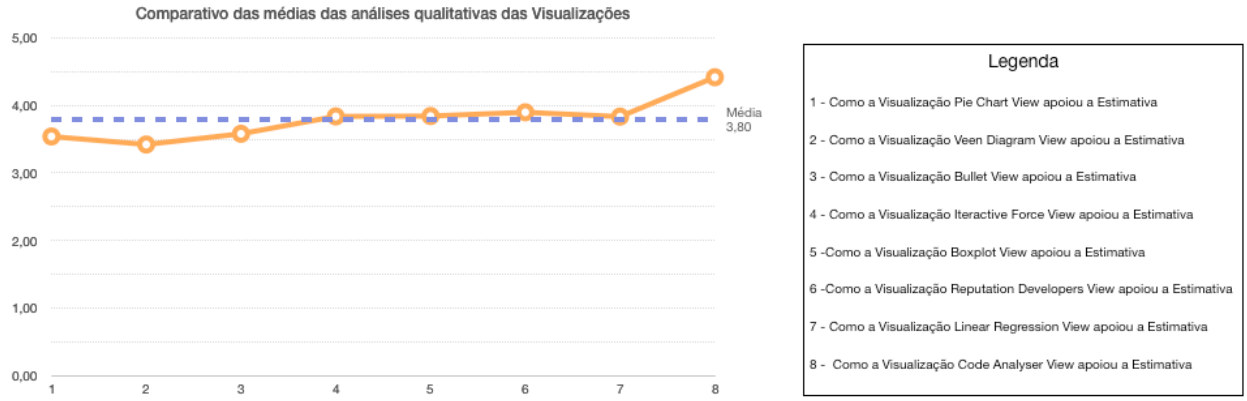


Figura 80: Comparativo de análise qualitativa das visualizações.

A Figura 81 apresenta um comparativo de análise qualitativa das visualizações por desenvolvedor em função de cada visualização proposta pelo *framework.* A linha tracejada apresenta a média geral dos participantes em função de cada visualização. Pode-se notar que na visualização *Code Analyser* (item 8 da legenda) foram destacados 4 dos 6 participantes apoiando a visualização e compreensão do software, estando as mesmas acima da média destacada na linha pontilhada. Analogamente, a visualização Linear Regression recebeu a mesma nota geral acima da média. As demais

visualizações mostraram-se equilibradas em função da média geral e da média de cada uma das técnicas.

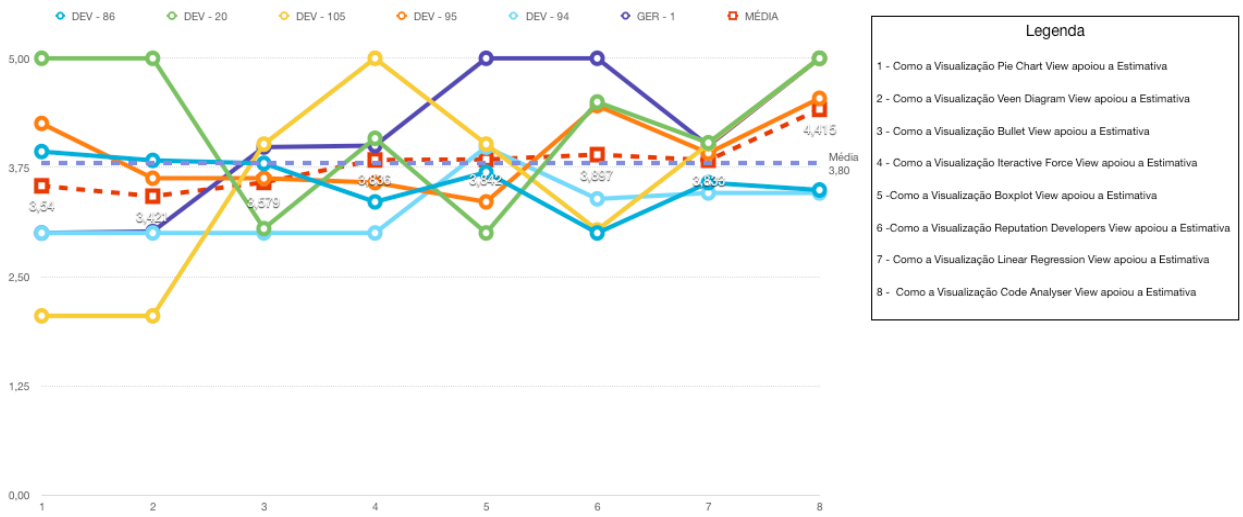


Figura 81: Comparativo de análise qualitativa das visualizações por desenvolvedor.

As análises realizadas tiveram como objetivo responder as duas questões elaboradas no GQM, a relembrar:

- Q1: O *GiveMe Effort* proporciona a equipe de desenvolvimento mecanismos para que se possa calcular a estimativa de esforço das atividades de manutenção de software?
- Q2: O *GiveMe Effort* apoia a visualização, compreensão e manutenção de softwares?

Os dados apresentados pelo experimento apresentam evidências de que é possível que o *framework GiveMe Effort* proporcione mecanismos para calcular a estimativa de esforço das equipes de desenvolvimento, apoiando as atividades de manutenção de software.

A segunda questão pode ser respondida pelas visualizações que foram viabilizadas pelo uso do *framework*, conforme abordado na seção 3 deste trabalho. Essas visualizações permitem, a cada requisição de mudança calculada, uma gama de informações úteis que podem auxiliar na tomada de decisão por parte do usuário. Essas visualizações apoiam a compreensão do código a ser tratado na requisição de mudança e fornece métricas e comparações em relação às demais requisições de mudança comparadas. Isso pode ser notado, por exemplo, na Visualização *Code Analyser* que permitiu que os participantes pudessem compreender as interligações entre as

requisições de mudança e o código fonte afetados, bem como, observar que requisições de mudança interagem entre si utilizando o mesmo código fonte.

Com base nos objetivos definidos e nas questões respondidas pelo GQM, pode-se rejeitar a hipótese nula e aceitar a hipótese alternativa de que a utilização da *GiveMe Effort* apresentou indícios no apoio a estimativa de esforço em um ambiente baseado em múltiplas visões interativas apoiando a manutenção e compreensão do software.

Como lições aprendidas, pode-se notar, que como boas práticas em relação ao uso do *framework* as visualizações Code Analyser, Linear Regression View, Box Plot View e Reputation Developers View foram melhores aproveitadas e utilizadas pelos usuários, auxiliando na estimativa de esforço em relação às demais visualizações apresentadas.

5.6 AMEAÇAS À VALIDADE

As ameaças à validade dessa prova de conceito são relacionadas à base de dados utilizadas e à metodologia de desenvolvimento usada pela equipe. Utilizou-se uma base de dados, durante todo o experimento, que possuía os dados de requisições de mudanças muito bem documentados, bem como cada requisição e o tempo real gasto na mesma, foi armazenada no banco de dados da empresa. A aplicação desse estudo em bases mal documentadas e nas quais o tempo gasto efetivo em cada desenvolvimento não é armazenado, compromete a aplicação das técnicas apresentadas.

O tempo real gasto pelo desenvolvedor em cada requisição de mudança, historicamente armazenados, serve como base para os cálculos das estimativas, visto que, a seleção de tais requisições que contenham a mesma similaridade e os tempos reais gastos, permite computar a estimativa final, calculada pelo framework baseando-se nas técnicas apresentadas nesse trabalho.

Não houve nenhum tipo de teste em relação à aplicação dessas técnicas a uma empresa que utiliza outra metodologia de desenvolvimento que não seja ágil, como por exemplo o SCRUM (SCHWABER, 2015), adotado pela empresa parceira que forneceu os dados para o experimento e trabalho desenvolvido

Não se testou a extensibilidade do *framework* com a inclusão de novos módulos como de Reputação ou a troca de uma técnica, visto que a arquitetura modular permite tão ação.

Não foi objetivo desse trabalho, tratar a segurança dos dados ou acesso da aplicação, bem como, a utilização por parte de um desenvolvedor ao tentar utilizar um módulo no contexto de outra aplicação.

Também não se tratou a velocidade de processamento em seus cálculos ou carga de base de dados inicial feito pelo *framework*.

5.7 CONSIDERAÇÕES FINAIS SOBRE O CAPÍTULO

Este capítulo apresentou uma avaliação da solução por meio de uma prova de conceito em uma base real, com o objetivo de verificar o apoio dado pelo *GiveMe Effort* nas estimativas de esforço das atividades de manutenção de software. Foram considerados as estimativas de esforço realizadas pela equipe da empresa parceira, bem como as geradas pelo *framework*, como prova de sua viabilidade e uso. Foram analisados os resultados advindos do uso do *framework GiveMe Effort* em um cenário real de desenvolvimento. *GiveMe Effort* apoiou a estimativa de esforço e proveu visualização para uma melhor compreensão das atividades de manutenção e compreensão de software.

6. CONCLUSÕES

O *framework* proposto promove a estimativa de esforço para equipe de desenvolvimento Ágil, integrando-se ao *framework GiveMe Infra.* Através deste é possível à geração de visualizações que podem auxiliar as equipes de desenvolvimento, inclusive geograficamente distribuídas, no planejamento das estimativas de esforço aplicadas às atividades de manutenção e compreensão de software.

A arquitetura do *framework GiveMe Effort* também tem o objetivo de apoiar as equipes de desenvolvimento no planejamento dos ciclos de desenvolvimento e manutenção e compreensão de software, auxiliando no dimensionamento do esforço gasto para cada atividade dentro de um *Sprint*, por meio de técnicas de similaridade em um ambiente integrado de múltiplas visualizações (AIMV). O tempo total de diversas atividades pode ser somado e adicionado ao planejamento da equipe, sendo disponibilizada a consulta de forma histórica.

O *framework* de apoio à estimativa de esforço, disposto neste trabalho, apresenta-se como suporte ao planejamento das atividades de manutenção de software, visto que o referido planejamento é essencial para as equipes de desenvolvimento. O resultado almejado com os ganhos, relativos às informações obtidas pelo uso do *framework*, são a redução dos problemas no ciclo de manutenção, que impactam diretamente na entrega das demandas ao cliente, seja pelo atraso na conclusão ou pela qualidade do produto final que está sendo entregue ao cliente.

Como contribuições, esta pesquisa apresentou técnicas de estimativa de esforço integradas à manutenção e compreensão de software em um AIMV. A união de diferentes técnicas, que permitiram buscar em uma base histórica dados similares e derivar dos mesmos a estimativa de esforço, caracteriza também uma contribuição. As visualizações criadas, se propuseram a contribuir para que o usuário do *framework,* pudesse compreender melhor as manutenções e analisar as correlações entre elas. Adicionalmente, auxiliaram na tomada de decisão durante o processo de estimativa de esforço nas manutenções corretivas e adaptativas do ciclo de desenvolvimento de um software.

Como contribuição também para a comunidade de desenvolvimento, foram implementadas algumas técnicas na linguagem de desenvolvimento Java, como a

distância de Levenshtein, Regressão Linear e a normalização e cálculo de reputação baseado em dados históricos. Foram ainda implementadas nessa mesma linguagem, o cálculo de *outliers,* primeiro e terceiro quartil e se integrou e ajustou o *framework* para cálculo de similaridade utilizando a técnica Sumo Metrics. Visões foram adaptadas e criadas em *Java Script*. Por fim, os códigos foram desmobilizados no site do projeto para que pesquisas futuras possam se beneficiar dos resultados obtidos.

6.1 DIFICULDADES ENCONTRADAS

Durante todo a trajetória projeto, várias dificuldades foram encontradas e tentou-se saná-las para que se pudesse chegar a um projeto mais consistente.

Um tempo razoável foi gasto no levantamento do tipo de pesquisa que seria realizado na área de Manutenção e Evolução de software, até que um nicho de estudo surgiu, diante da necessidade de estimar esforço, levantado pela Empresa parceira.

Adicionalmente, a definição da plataforma a ser adotada caracterizou um desafio, visto que o projeto em curso (GiveMe Infra) estava sendo feito voltado para a IDE de desenvolvimento Eclipse. Diante dos requisitos existentes decidiu-se então pela plataforma Web e por isso os cálculos de similaridades ficaram em Java Desktop e todas as visualizações, presentes no *framework,* estão voltados para essa plataforma.

Por fim, a condução do experimento revelou-se um desafio devido à necessidade de utilizar um contexto real de utilização.

6.2 TRABALHOS FUTUROS

Como trabalhos futuros um estudo mais avançado, considerando-se o suporte à reputação, pode ser realizado para que o mesmo seja utilizado no mecanismo de cálculo da estimativa de tempo de atividades de manutenção de software. Este suporte deve considerar a reputação do desenvolvedor e seus dados históricos, inclusive em ambientes distribuídos.

Outras técnicas de similaridade não contempladas pela revisão sistemática feita para o presente trabalho podem ser adicionadas ao *framework*, buscando-se assim aumentar a precisão e as opções para seus usuários.

Cabe também avaliar a possibilidade de migração do *framework* para Web e utilizar *Cloud Computing* para torná-lo disponível à comunidade de desenvolvedores.

Além disso, estudos adicionais são necessários utilizando-se bases de dados diferentes daquela que foi disponibilizada pela empresa parceira, bem como, a adequação das métricas para outras metodologias de desenvolvimento que não estejam em um contexto de desenvolvimento Ágil. Neste contexto, artefatos podem ser incluídos para as análises de forma histórica, tais como: protótipos de interface e artefatos gerados na engenharia de requisitos, que podem ser processados e gerar parâmetros para a estimativa de esforço.

Pretende-se criar mecanismos de segurança de acesso a ferramenta, bem como dos dados por ela apresentados e processados.

Um novo mecanismo de estimativa de esforço distribuída seria interessante ser adicionado ao trabalho, visto que vários usuários, em lugares geograficamente distribuídos, poderiam contribuir com suas estimativas e cruzar informações e estimativas encontradas por cada um deles, levando a um consenso em mais de um usuário do sistema sobre a estimativa de tempo final. Pode-se também utilizar esse recurso como um mecanismo de checagem de valores entre a equipe de desenvolvimento e o responsável pelas funcionalidades(*Product Owner*).

A integração com outros artefatos para apoiar as visualizações e, como resultado, apoiar a tomada de decisão na estimativa de esforço. Por exemplo, como identificar qual usuário tem um perfil mais adequado para uma determinada atividade e quais as competências do mesmo para o projeto, ou a possibilidade de realocar um desenvolvedor de outro projeto, que seja mais competente para a realização de uma determinada tarefa.

Adicionalmente, poderia armazenar no *framework* as estimativas anteriores feitas pelos desenvolvedores ao longo do ciclo de vida do software, anteriores à utilização da solução apresentada nesse trabalho. Isso permitiria uma melhor calibração da ferramenta e refinamento das estimativas realizadas pelo *framework*.

Por fim, existe a possibilidade de incluir mecanismos de integração entre várias IDE's de desenvolvimento, tais como: Eclipse, NetBeans e Intellij com o *framework GiveMe Effort.*

REFERÊNCIAS

BASILI, V. R. C., GIANLUIGI, H. R. D. 1994 **The Goal Question Metric Approach. Chapter in Encyclopedia of Software Engineering,** Wiley.

CARNEIRO, G., CONCEIÇÃO C. F., DAVID, J. M. N. 2012. **A Multiple View Environment for Collaborative Software Comprehension**. ICSEA: The Seventh International Conf. on Software Engineering Advances, Portugal, pp. 15-21.

CORDEIRO, J., DIAS, G., & BRAZDIL, P. 2007. **A metric for paraphrase detection.** In Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on (pp. 7-7). IEEE.

D'AMBROS M., LANZA, M. 2010 **Distributed and Collaborative Software Evolution Analysis with Churrasco.** Experimental Software and Toolkits (EST 3): A special issue of the Workshop on Academic Software Development Tools and Techniques (WASDeTT), pp. 276–287.

DALL'OGLIO, P., da SILVA, J. P. S., & PINTO, S. C. C. 2010. **ChangeMan: Um Sistema Multi-Agente para Gestão da Mudança de Requisitos com suporte à Rastreabilidade e Análise de Impacto**. In VI Simpósio Brasileiro de Sistemas de Informação SBSI 2010. Marabá/PA, pp. 61-72.

FEDOTOVA, O.; TEIXEIRA, L.; ALVELOS, H. 2013 **Software Effort Estimation with Multiple Linear Regression: Review and Practical Application.Journal of Information Science and Engineering**, v. 29, n. 5, pp. 925-945.

FRANCO, N. B. 2006 **Cálculo Numérico**. Pearson.

FUKS, H., Raposo, A.B., GEROSA, M.A. 2003 **Do Modelo de Colaboração 3C à Engenharia de Groupware.** Simpósio Brasileiro de Sistemas Multimídia e Web– Webmidia, pp. 0-8.

GIT BOOK - http://git-scm.com/book Acesso em 01 ago. 2016.

JORGENSEN, M. (2004). **A review of studies on expert estimation of software development effort. Journal of Systems and Software**, 70(1), pp. 37-60.

LÉLIS, C. A. S. GiveMe Trace: **Uma ferramenta de apoio à rastreabilidade de software.** Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal de Juiz de Fora, 2014.

KITCHENHAM, B.A., CHARTERS, S. 2007. **Guidelines for performing systematic literature reviews in software engineering.** Tech. Rep. EBSE-2007-01, Keele University.

KEVIC, K., MULLER, S. C., FRITZ, T., & GALL, H. C. 2013. **Collaborative bug triaging using textual similarities and change set analysis.** Cooperative and Human Aspects of Software Engineering (CHASE), 6th International Workshop on, IEEE, pp. 17-24.

LIKERT, R. 1932 **A technique for the measurement of attitudes**. Archives of psychology, p. 140.

LEVENSHTEIN V. 1966 **Binary Codes Capable of Correcting Deletions, Insertions, and Reversals**, Soviet Physice-Doklady, 10:707-710.

MANTIS, https://www.mantisbt.org. Acesso em 01 ago. 2016.

MASSART, D. L., SMEYERS-VERBEKE, J. C. X., & SCHLESIER, K. 2005 **Visual presentation of data by means of box plots**. LC-GC Europe, 18: pp. 215-218.

MCCONNELL S. 2006 **Software Estimation Demystifying the Black Art**. Microsoft Press.

MIGUEL, M., ARAÚJO, M. A., DAVID, J. M. N. **Estimativa de tempo nas atividades de manutenção de software em métodos ágeis apoiadas por visualização de software** - Revisão Sistemática, PPGCC/UFJF. Disponível em: http://givemeeffort.tk - seção artigos e publicações. Acesso em 01 ago. 2016.

MIGUEL, M., TAVARES, J., DAVID, J. M. N. 2014. **Visualização nas Atividades de Evolução de Software no Processo de Manutenção Colaborativa - Revisão Sistemática**, PPGCC/UFJF. Disponível em: http://givemeeffort.tk - seção artigos e publicações. Acesso em 01 ago. 2016.

MOHAN, K., XU, P., CAO, L., & RAMESH, B. 2008. **Improving change management in software development: Integrating traceability and software configuration management**. Decision Support Systems, 45(4), 922-936.

MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. 2012 **Introduction to linear regression analysis.** John Wiley & Sons.

PAREDES, J., ANSLOW, C., & MAURER, F. 2014. **Information Visualization for Agile Software Development**. In Software Visualization (VISSOFT), 2014 Second IEEE Working Conference on. pp. 157-166.

PILATO, C. M., COLLINS-SUSSMAN, B., & FITZPATRICK, B. W. 2008 **Version control with subversion.** O'Reilly Media, Inc.

PIMENTEL, M., FUKS, H., LUCENA, C. J. 2008. **Um processo de desenvolvimento de sistemas colaborativos baseado no Modelo 3C: RUP-3C-Groupware**. Anais do IV Simpósio Brasileiro de Sistemas de Informação, pp. 37-47.

PONCIN, W., SEREBRENIK, A., VAN DEN BRAND, M. 2011 **Process mining software repositories**. Software Maintenance and Reengineering (CSMR), 2011 15th European Conference on. IEEE. pp. 5-14.

PRIKLADNICKI, R., AUDY, J. L. N. 2007 **Desenvolvimento Distribuído de Software**, CAMPUS – RJ.

RAJLICH, V. 2014. **Software Evolution and Maintenance**, FOSE 2014 Proceedings of the on Future of Software Engineering, pp. 133-144.

SANTANA, F. et al. 2011 **Xflow: An extensible tool for empirical analysis of software systems evolution.** In: Proceedings of the VIII Experimental Software Engineering Latin American Workshop, ESELAW (Vol. 11), pp. 57-66

SERRANO, N., & CIORDIA, I. 2005 **Bugzilla, ITracker, and other bug trackers. Software**, IEEE, 22(2), pp. 11-13.

SILVA, A. N. C., G. F. ZANIN, R. B. DAL POZ, A. P. MARTINS, E. F. O. 2012 **Propondo uma Arquitetura para Ambientes Interativos baseados em Múltiplas Visões.** II Workshop Brasileiro de Visualização de Software, Natal, RN, pp. 1 – 8.

SIRQUEIRA, T; MIGUEL, M. A.; DALPRA, H.; ARAÚJO, M. DAVID, J. M. **Application of Statistical Methods in Software Engineering: Theory and Practice**. Software: Practice and Experience, submetido em 06 de Junho de 2016. Disponível em: http://givemeeffort.tk - seção artigos e publicações. Acesso em 01 ago. 2016.

SOMMERVILLE, I. 2011 **Engenharia de Software**. 9ª Edição. Pearson Addison-Wesley.

STOREY, M. D., ČUBRANIĆ D., GERMAN, D, M. 2005. **On the use of visualization to support awareness of human activities in software development: a survey and a *framework***. Proceedings of the 2005 ACM Symposium on Software Visualization, pp. 193-202.

SCHWABER, K.; SUTHERLAND, J. 2015. **Scrum Guide**. Disponível em: https://www.Scrum.org/Scrum-Guide. Acesso em 01 ago. 2016.

SHIN, Sun-Joo. 1994 **The logical status of diagrams**. Cambridge University Press.

SVN - https://subversion.apache.org. Acesso em 01 ago. 2016.

TAVARES, J., DAVID, J. M. N., ARAÚJO, M. A. P., BRAGA, R. M., CAMPOS, F. C. A., CARNEIRO, G. F. **Uma Infraestrutura baseada em Múltiplas Visões Interativas para Apoiar Evolução de Software**. iSys: Revista Brasileira de Sistemas de Informação, v. 8, pp. 65-101, 2015.

TAVARES, J. F., David, J. M. N., Araújo, M. A. P., Braga, R., Campos, F. C. A., Carneiro, G. F. **GiveMe Views: a support tool to software evolution based on historical data analysis.** In: XI Brazilian Symposium on Information Systems, 2015,

Goiânia - GO. SBSI 2015 Proceedings of the annual conference on Brazilian Symposium on Information Systems: Information Systems: A Computer Socio-Technical Perspective. Porto Alegre - RS: Brazilian Computer Society, 2015. v. I. p. 55-62.

WERNER, C., MURTA, L., SCHOTS M., MAGDALENO A., SILVA M., CEPEDA R., VAHIA C., 2011, **EvolTrack: A Plug-in-Based Infrastructure for Visualizing Software Evolution**. I Workshop Brasileiro de Visualização de Software, CBSoft, São Paulo, Brazil, pp. 1-8.

ZIA, Z. K., SHAHID K. T. 2012 **An effort estimation model for agile software development.** Advances in Computer Science and its Applications 2.1, pp. 314-324.

ØSTVOLD, K.M.; HAUGEN, N.C.; BENESTAD, H.C. 2007. **Using Planning Poker for combining expert estimates in software projects.** Australian Software Engineering Conference (ASWEC, 2007), Journal of Systems and Software, Volume 81, Issue 12, pp. 2106–2117.