Combinando *Planning Poker* e Aprendizado de Máquina para estimar esforço de software

Doglas A. Finco[1], Laudelino C. Bastos[1], Adolfo G. S. S. Neto[1]

[1]Departamento de Informática (DAINF) - Universidade Tecnológica Federal do Paraná (UTFPR) – Curitiba, PR - Brasil

doglas.andref@gmail.com, bastos@dainf.ct.utfpr.edu.br, adolfo@utfpr.edu.br

Abstract. *Estimating software effort is critical to organizations. One of the main practices is Planning Poker, however your data is not saved for future estimation. The use of machine learning (ML) is growing and so we propose a combination of Planning Poker and ML, called ML Planning Poker. It consists of using Planning Poker and a tool containing a built-in ML model. We evaluated the proposal in an academic environment and with IT professionals. In both scenarios the reports were favorable to use ML Planning Poker. The combination presented less errors when compared to the original Planning Poker, more security bringing similar tasks and reducing uncertainty in estimates.*

Resumo. *Estimar esforço de software é crítico às organizações. Uma das principais práticas é o Planning Poker, porém seus dados não são salvos para estimativas futuras. O uso de aprendizado de máquina (AM) vem crescendo e assim propomos uma combinação do Planning Poker e AM, denominada ML Planning Poker. Consiste em utilizar o Planning Poker e uma ferramenta desenvolvida contendo um modelo de AM embutido. Avaliamos a proposta em ambiente acadêmico e com profissionais de TI. Em ambos os cenários, relatos foram favoráveis ao uso da ML Planning Poker. A combinação apresentou menos erros se comparada ao Planning Poker original, mais segurança trazendo tarefas similares e reduzindo a incerteza nas estimativas.*

1. Introdução

A estimativa de esforço de desenvolvimento de software é o processo de prever o esforço necessário para desenvolver um software [Wen et al. 2012, Tissot et al. 2015]. [Ziauddin e Zia 2012] afirmam que estimar esforço em projetos de software é um ponto crítico para toda organização. A subestimativa pode resultar em pressão de tempo, comprometendo o desenvolvimento funcional. Da mesma forma, a superestimativa pode culminar em orçamentos não competitivos [Tronto et al. 2008].

Mesmo com o surgimento e aplicação de técnicas de estimativas, o percentual de imprecisão ainda é grande. O tema foi abordado por [Eman e Koru 2008] num estudo entre os anos 2005 e 2007 considerando falhas gerais (projetos cancelados ou entregues com desempenho malsucedido), os números indicaram que 26% a 34% dos projetos de Tecnologia da Informação (TI) ainda falham. Os autores ressaltam que, apesar de muitos anos de pesquisa, a habilidade de estimar continua sendo um desafio aos projetos de TI,

pois os profissionais não usam as melhores ferramentas e técnicas ou porque estas exigem melhorias antes de serem usadas efetivamente. No mesmo sentido, uma pesquisa conduzida por [Bloch et al. 2012] em colaboração com a Universidade de Oxford identificou que metade dos grandes projetos de TI - definidos como aqueles com preços superiores a U$$ 15 milhões – estouram seus orçamentos. Em média, grandes projetos de TI ficam 45% acima do orçamento e 7% acima do tempo, ao mesmo tempo que fornecem 56% menor valor agregado do que o previsto.

De acordo com [Tronto et al. 2008] as técnicas ou métodos de estimativas de esforço de software podem ser classificados em três categorias: opinião especializada, modelos algorítmicos e aprendizado de máquina. No contexto da opinião especializada o *Planning Poker* apresenta-se como uma das principais práticas. [Cohn 2005] afirma que o *Planning Poker* consiste da combinação de especialistas e separação das partes envolvidas em uma abordagem que apresenta resultados confiáveis rapidamente. Esta forma de estimar considera a experiência dos componentes da equipe, ou seja, decisões baseadas em conhecimento empírico. A seguir são descritas as etapas da prática:

1. Cada membro recebe um baralho de cartas. Cada carta tem um valor de estimativa válida. As estimativas válidas possuem relação com sequência de Fibonacci e geralmente contém os valores 0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, e 100;
2. Para cada tarefa, um moderador que geralmente é um representante do cliente lê a descrição da tarefa. O moderador responde todas as perguntas que os membros da equipe têm relacionado à tarefa;
3. Após as perguntas serem respondidas, cada estimador seleciona de maneira privada uma das cartas que representa sua estimativa. A carta que representa a estimativa escolhida é baseada no conhecimento do estimador. As cartas não são mostradas até que todos os membros tenham feito suas escolhas;
4. A partir do momento que todos os membros realizaram suas escolhas, as cartas são viradas simultaneamente para que todos possam ver cada estimativa;
5. Nesta etapa, as estimativas podem divergir significativamente. Se isso ocorrer, os estimadores que escolheram os valores mais alto e mais baixo explicam os fatores analisados e os motivos para a escolha daquela estimativa;
6. O grupo discute sobre a tarefa e suas estimativas por alguns minutos. Após a discussão cada membro seleciona novamente uma carta. Mais uma vez a carta é mantida em sigilo até que todos os membros tenham feito suas escolhas. Isso se repete até que haja um consenso.

Em diversos casos as estimativas convergem na segunda rodada e raramente ocorrem mais de três rodadas. O objetivo é convergir para uma estimativa única que seja usada na tarefa. Não é necessário que todas as estimativas sejam iguais, dependerá do moderador, pois o objetivo não é a precisão absoluta, mas sim razoabilidade. Um ponto positivo da técnica é que a medida em que os membros justificam as suas estimativas ocorre a disseminação do conhecimento entre as pessoas envolvidas [Cohn 2005].

Porém, segundo [Tissot et al. 2015] as informações geradas pelo debate no *Planning Poker* não são guardadas devido à informalidade aparente do método e como esse conhecimento se perde, não há como aproveitá-lo em estimativas futuras. A

utilização do *Planning Poker* permite a troca de conhecimento entre estimadores e diminui o excesso de otimismo que geralmente ocorre em julgamentos individuais, no entanto, o fato das informações não serem guardadas, a subjetividade, outros aspectos humanos e a falta de experiência com tarefas semelhantes podem resultar em falhas.

Conforme [Dantas et al. 2018], desde 2014 a comunidade científica vem sendo bastante ativa na área de estimativa de esforço no desenvolvimento ágil de software, porém, o tema continua sendo desafiador e objeto de mais estudos, dada a dificuldade em encontrar soluções precisas para o problema. Os autores reforçam em sua revisão sistemática que uma quantidade significativa de novos trabalhos tem usado técnicas de Inteligência Artificial (IA) ou Aprendizado de Máquina (AM) para apoiar a estimativa no desenvolvimento ágil, o que contribuiu para uma melhor precisão de estimativa.

A comunidade de pesquisa propôs diversos modelos e técnicas para alcançar alta precisão na previsão de esforço. Porém, apesar do grande número de estudos, não há consenso que determine o melhor método único, sendo assim, a combinação de técnicas para estimar esforço vem sendo explorada. Na Revisão Sistemática realizada por [Idri et al. 2016] é defendida a necessidade de superar as fraquezas das técnicas únicas originando o agrupamento das mesmas. Além disso, a revisão concluiu que técnicas utilizadas em conjunto geralmente são mais precisas que técnicas individuais.

Perante as possibilidades de combinar métodos para estimar esforço, levantamos a hipótese de que combinando *Planning Poker* com AM pode ser uma alternativa interessante culminando na melhoria do processo de estimativas. O *Planning Poker* valoriza o fator humano e o AM atua no apoio à tomada da decisão trazendo dados históricos. Diante do cenário discutido, este estudo objetiva descrever a combinação do *Planning Poker* com AM, nomeada *ML Planning Poker* e avaliar se a proposta interfere no processo de estimativas de esforço de software.

2. *ML Planning Poker*

A *ML Planning Poker* consiste da combinação do *Planning Poker* juntamente de um modelo de AM construído e embutido numa ferramenta implementada, servindo de apoio para as equipes de desenvolvimento nas estimativas. Conforme representado na Figura 1(a), o processo inicia-se com a preparação da equipe para a aplicação do processo de estimativas das tarefas da *sprint*, com participação do moderador, equipe de desenvolvimento e a ferramenta atuando similarmente a um integrante do time.

A equipe recebe a tarefa a ser estimada e descrita pelo moderador. Cada membro define sua carta do *Planning Poker* e o moderador insere a tarefa na ferramenta. Todos os membros trazem suas estimativas simultaneamente, inclusive a ferramenta, conforme representado na Figura 1(b). Complementarmente, a ferramenta apresenta uma lista com tarefas semelhantes realizada em estimativas anteriores como maneira de justificar-se (Figura 4). Perante o debate sobre as estimativas levantadas, a ferramenta não conseguirá alterar sua escolha, haja vista que não é capaz de mudar de predição via debate, mas os membros da equipe continuam esclarecendo sobre a tarefa até finalizar o processo do *Planning Poker* e definição da estimativa. O processo se repete para toda tarefa a ser estimada. Após a definição da estimativa de cada tarefa existe um campo na ferramenta que armazena o tempo estimado servindo para fins de registro e análises posteriores quanto a efetividade do processo de estimativas.

(a) Descrição da tarefa pelo moderador. (b) Estimativa simultânea dos membros.

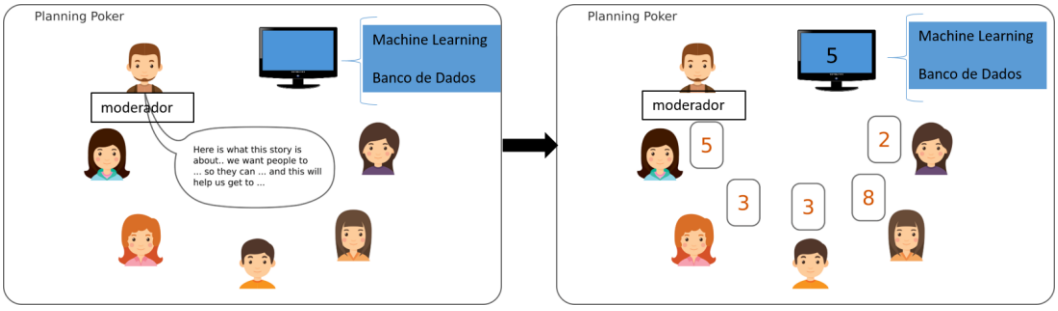

Figura 1. Esboço do projeto na prática

A Figura 2 apresenta o diagrama da *ML Planning Poker*, construído na ferramenta Bizagi[1] que utiliza o padrão BPMN (*Business Process Model and Notation*) e por meio das raias facilita a visualização do responsável de cada ação no processo.

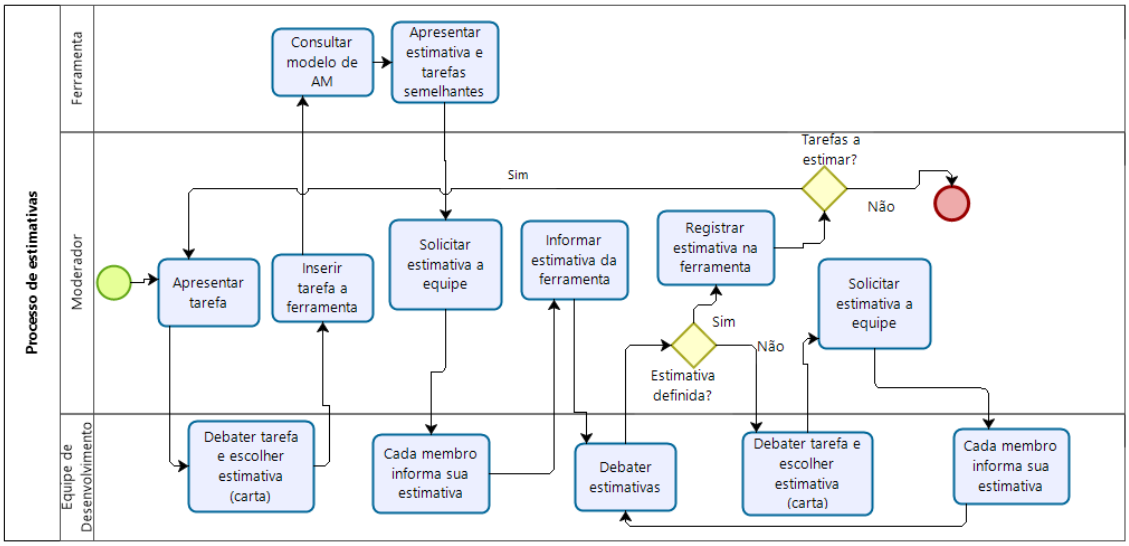

Figura 2. Etapas da *ML Planning Poker*

Note que o informe da estimativa definida pela ferramenta ocorre posteriormente ao informe da estimativa dos participantes, isso se dá buscando evitar que os estimadores sejam influenciados pelos resultados da ferramenta.

Defendemos a combinação perante o contexto em que a troca de conhecimento entre os integrantes da equipe se mantenha e os resultados trazidos através do *Planning Poker* continuem sendo valorizados. Mas também, o AM também seja consultado como um artefato complementar ao processo, gerando assim uma combinação que obtenha o melhor do fator humano juntamente a análise de dados históricos. Quando ocorre disparidade entre os resultados do *Planning Poker* e AM é por meio do debate e rodadas posteriores do *Planning Poker* que se define a estimativa.

[1] https://www.bizagi.com/pt/plataforma/modeler

2.1. Construção da Ferramenta

Para a implementação da ferramenta optamos pelo uso do Framework Django[2]. Inicialmente foram criadas as funcionalidades de cadastro, login, manter tarefas e vínculo dos membros às equipes. Num segundo momento, pós criação do modelo de AM descrito na seção 2.2, construímos a funcionalidade de buscar estimativa e finalmente a aplicação foi hospedada para uso. Optamos pela utilização do SGBD (Sistema Gerenciados de Banco de Dados) PostgreSQL, o padrão utilizado na ferramenta de hospedagem escolhida. A ferramenta foi nomeada TarEst[3], foi hospedada na plataforma gratuita Heroku[4] e o código pode ser acessado publicamente[5].

A funcionalidade a seguir trata-se da guia Minhas Tarefas, apresentando a lista das tarefas cadastradas pelo usuário (Figura 3). Podem ser visualizadas também as tarefas finalizadas nos últimos 30 dias, finalizadas e para fazer, bem como as opções de adição, busca, edição/estimar e remoção de tarefas.

Figura 3. Guia Minhas Tarefas

A funcionalidade Estimar Tarefa aborda resultados do processo de estimativas (Figura 4). Os resultados trazidos pela funcionalidade de estimativas são de acordo com o Modelo de ML construído na seção 2.2.

Perceba que nesta funcionalidade, com base nas variáveis descritivas (verbo inicial, substantivo, categoria e subcategoria) o modelo de ML tem a capacidade de estimar quanto tempo é necessário para realizar a tarefa. Como forma de argumentar sobre a estimativa definida, a ferramenta traz tarefas semelhantes mantidas numa base histórica, servindo de apoio a estimativa atual. Assim, o estimador que nunca realizou uma tarefa semelhante ou não lembra o esforço necessário para a tarefa específica, tem dados que possam ajudá-lo na tomada de decisão da estimativa.

---

[2] https://www.djangoproject.com/

[3] https://tarest.herokuapp.com/

[4] https://www.heroku.com/

[5] https://github.com/Doglas/PesquisaMestrado/blob/master/Tarest.rar

Figura 4. Funcionalidade Estimar Tarefa

2.2. Construção do Modelo de AM

Quando definida a utilização do AM no processo, sabíamos da necessidade em encontrar *datasets* em torno do tema, pois sem dados não existe modelo de AM. Iniciamos as buscas visando encontrar *datasets* de tarefas de software. O que dificultava era a especificidade de ter um grande número de tarefas de software com características e tempo realizado. Definimos uma *string* de busca denominada: "*dataset software task estimates*" que retornou uma base de dados com possibilidade de uso na pesquisa. O título da matéria contendo os dados chama-se: **Procurados: 99 conjuntos de dados de estimativa de esforço[6].** O título em si define a dificuldade em encontrar *datasets* públicos com estimativas de esforço de software, realizando um apelo para que mais conjuntos de dados sejam compartilhados.

O *dataset* definido foi traduzido pois a proposta foi aplicada em português e estudamos as colunas existentes, buscando perceber quais poderiam ser úteis. Exploramos os tipos de dados existentes (texto, inteiros ou decimal), além de verificarmos dados duplicados, faltantes, entre outros. Após análises, decidimos utilizar as seguintes colunas para características (*features*) descritivas das tarefas:

- Descrição: uma breve descrição de texto da ação necessária para concluir a Tarefa, projetada para ser significativa para o cliente sempre que possível.
- Categoria: um identificador que categoriza uma Tarefa de gerenciamento, operacional ou de desenvolvimento.
- Subcategoria: um tipo específico de categoria pai, por exemplo: aprimoramento de sistema, documentação, erro, gestão, suporte, lançamento, entre outras.

[6] http://shape-of-code.coding-guidelines.com/2019/01/10/wanted-99-effort-estimation-datasets/

A característica alvo (*target*) ficou definida a coluna Horas Atuais, que retorna um valor decimal equivalente ao total de horas para realizar a Tarefa.

Após a limpeza inicial de dados, percebemos que a coluna descrição necessitaria de processamento de linguagem natural para ser compreendida. Deste modo, alternativamente optamos em realizar o processo de engenharia de *features*, ou seja, sobre a descrição retiraríamos informações que pudessem contribuir com o processo de padronização das tarefas. Assim, criamos a coluna Verbo Inicial, contendo o verbo inicial de cada tarefa. O *dataset* ainda trazia muitas tarefas com *features* similares, mas com estimativas discrepantes. Então, optamos em definir o substantivo da tarefa através da descrição, que na grande maioria das vezes acompanhava o verbo inicial. A criação desta nova *feature* permitiu maior diferenciação entre as tarefas. Estas etapas permitiram chegar ao *dataset* final[7] utilizado.

Realizada a preparação dos dados, definimos algoritmos consolidados de AM para aplicação. Dentre tantos optamos por um modelo baseado em informação, a Árvore de Decisão; um em similaridade k-NeighborsClassifier; um em probabilidade, o Naive Bayes e por fim o XGBoost, modelo baseado em Árvore de Decisão e que utiliza estrutura de Gradient boosting. Dentre os modelos, a Árvore de Decisão obteve melhores resultados e utilizamos. A Figura 5 mostra o *classification_report* do modelo.

```
print(classification_report(y_test, previsoesdectree))
```

	precision	recall	f1-score	support
0	0.62	0.63	0.62	110
1	0.56	0.55	0.56	111
2	0.45	0.47	0.46	53
3	0.41	0.38	0.39	39
accuracy			0.54	313
macro avg	0.51	0.51	0.51	313
weighted avg	0.54	0.54	0.54	313

Figura 5. Resultados do Modelo

Os resultados do *classification_report* indicam para um desempenho razoável do modelo se considerarmos o *weighted avg* de 0,54. Quanto mais próximo de 1 é o desempenho, maior a assertividade do modelo.

Em torno da qualidade razoável do modelo, percebemos situações no *dataset* que poderiam resultar em problemas. Dentre eles, destacamos que se trata da exploração de dados num período de 10 anos, sobre vários projetos de software, que contaram com 22 membros diferentes. Por referir-se aos anos 2004 e 2014 a realidade da época não exigia uma padronização tão grande das tarefas. O ato de ter poucas colunas (variáveis descritivas) para diferenciar as tarefas dificultou o aprendizado dos modelos, exigindo a criação de *features* para que resultados aceitáveis fossem atingidos. O modelo completo com as etapas apresentadas e descritas está disponível para acesso[8].

[7] https://github.com/Doglas/PesquisaMestrado/blob/master/CSVTarefas.csv

[8] https://github.com/Doglas/PesquisaMestrado/blob/master/MLMestradoProducao.ipynb

3. Avaliação

A avaliação ocorreu de maneira qualitativa e quantitativa. A dificuldade de inserção nas empresas impediu a avaliação em ambiente real. Deste modo, realizamos avaliação em sala de aula com estudantes de graduação e com profissionais de TI.

3.1. Ambiente Acadêmico

Para a avaliação em ambiente acadêmico utilizamos duas turmas de estudantes na disciplina de Projeto e Análise de Sistemas nos cursos de graduação de Sistemas da Informação e Engenharia da Computação de uma universidade. Participaram da pesquisa 32 estudantes divididos em 9 grupos, através da abordagem antes/depois. Inicialmente os estudantes aplicaram o *Planning Poker* original na metade das tarefas estimadas e depois estimaram outra metade das tarefas utilizando a *ML Planning Poker*, ao final da implementação das tarefas foi armazenado na ferramenta o tempo realizado de cada tarefa para a análise dos resultados (Tabela 1).

Tabela 1. Abordagem antes/depois

PLANNING POKER ORIGINAL VERSUS ML PLANNING POKER				
	Planning Poker original		*ML Planning Poker*	
TAREFAS	37	100%	37	100%
ASSERTIVIDADE	9	24,3%	9	24,3%
SUBESTIMATIVA	7	18,9%	13	35,1%
SUPERESTIMATIVA	21	56,7%	15	40,5%
EM TORNO DAS SUBESTIMADAS OU SUPERESTIMADAS				
TAREFAS	28	100%	28	100%
DIFERENÇA <=1	11	39,2%	16	57,1%
DIFERENÇA > 1	17	60,7%	12	42,8%

Na comparação apresentada, visualizamos a assertividade (igualdade no tempo estimado versus realizado) em 24,3% em ambas as abordagens. O *Planning Poker* original teve um número maior de tarefas superestimadas (56,75%) em relação ao total. Por sua vez, a *ML Planning Poker* teve equilíbrio entre tarefas subestimadas (35,1%) e superestimadas (40,5%). Em torno da análise das tarefas com estimativa incorreta, percebemos que a proposta teve melhor resultado, haja vista que, 57,1% das tarefas teve diferença máxima de 1 hora entre estimado e realizado em comparação a 39,2% do *Planning Poker* original. Deste modo também, 60,7% das tarefas do *Planning Poker* original teve erro maior de 1 hora em comparação a 42,8% da *ML Planning Poker*.

Após a realização do processo de estimativas com o apoio da *ML Planning Poker*, aplicamos um questionário com os participantes com intuito de compreender suas percepções. No que diz respeito a Q4: “Qual é sua percepção quanto a afirmação: A combinação do *Planning Poker* com a proposta contribui no processo de estimativas. ”, buscamos identificar se os participantes perceberam contribuição da proposta na melhoria do processo de estimativas; os resultados encontram-se na Figura 6.

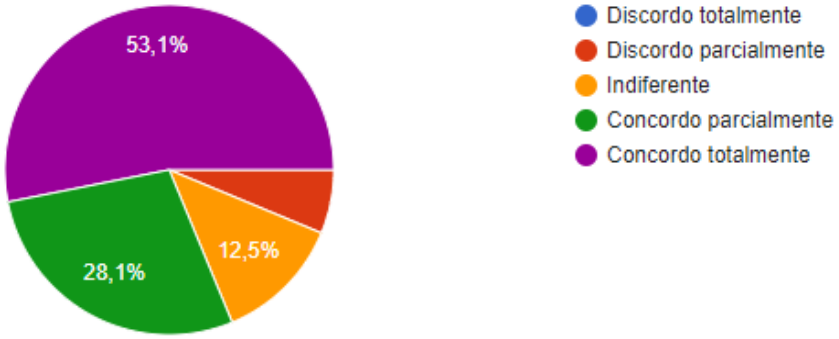

Figura 6. Resultados da Q4 (Acadêmicos)

Nesta questão percebemos que 81,2% concordam que a *ML Planning Poker* contribui; destes, 53,1% totalmente, enquanto 28,1% parcialmente. Isso permite considerar que a combinação de *Planning Poker* e AM apoia o processo e traz uma percepção positiva quanto ao uso, direcionando para uma interferência positiva da combinação das técnicas nas estimativas de esforço de software.

3.2. Profissionais de TI

Nesta etapa convidamos profissionais de TI via redes sociais e Google Forms, dos quais 20 participaram. Organizamos uma reunião via web conferência, os participantes conheceram a ideia, o fluxo da proposta, o esboço do projeto na prática e posteriormente a ferramenta desenvolvida. Apresentamos as funcionalidades, bem como o processo de utilização, sendo solicitado aos participantes a criação de uma conta e utilização da TarEst para experimentação. Por fim, os participantes responderam a um questionário.

No que diz respeito a Q5: “Quais são as percepções positivas quanto ao uso da proposta? ”, algumas das respostas são apresentadas na Tabela 2.

Tabela 2. Percepções positivas a proposta (Profissionais de TI)

Resposta
Todos pontos apresentados são muito positivos mas o que mais me chamou a atenção é trazer um histórico de tarefas semelhantes, isto ajuda muito principalmente quando a média da votação dos participantes tiver uma divergência muito diferente da estimada por machine learning. Em casos como estes agiliza o processo de debater as tarefas que podem ser mais complexas e não perder tempo em tarefas mais simples.
Pode agregar muito valor, principalmente para times que tem uma grande variedade na experiência dos integrantes, como um sênior com anos de experiência, e um junior recém contratado. A ferramenta pode dar um "norte" para os integrantes, independente do nível de experiência.
Tendo uma máquina que leva em consideração apenas o histórico já desenvolvido, as estimativas muito "fora da curva" tendem a diminuir... Os participantes do planning podem se esquecer de uma tarefa semelhante desenvolvida tempos atrás que poderia ser utilizada como base nessa estimativa, nesses casos a ferramenta iria auxiliar bastante.
A proposta é muito boa, considerando que muitas equipes de desenvolvimento utilizam o Planning Poker, existem casos em que acontecem divergências entre decisões de tempo, isso podendo acontecer por N fatores, entre tanto, utilizando o Aprendizado por Máquina, pode ajudar a equipe em novos debates e discussões sobre os motivos de tempo escolhido por cada integrante, levanto para um caminho mais "feliz". Isso pelo fato de ter uma IA como "integrante", que possui os dados históricos de cada tarefa trazendo uma perspectiva diferente para a equipe. Esse é um trabalho perfeito para uma IA, já que nós não somos capazes de lembrar de todas as tarefas e tempo levado sobre elas.

4. Considerações Finais

O presente trabalho descreveu a *ML Planning Poker*, combinação do método *Planning Poker* e Aprendizado de Máquina (AM) e avaliou se a combinação interfere no processo de estimativas de esforço de software. A avaliação com estudantes resultou na assertividade (igualdade no tempo estimado versus realizado) semelhante nas tarefas estimadas usando o *Planning Poker* original e a proposta. Porém, dentre as tarefas com erro de estimativas, a *ML Planning Poker* apresentou melhor proximidade entre estimado/realizado na maioria das tarefas. Além disso, dos estudantes participantes, 81,2% concordam que a abordagem contribui com o processo de estimativas.

Os profissionais de TI perceberam benefícios ao processo, defendem que o AM proporciona um "norte" aos integrantes. Reforçaram também o problema do esquecimento de tarefas muito antigas o que dificulta a estimativa atual, sendo que a *ML Planning Poker* auxilia, pois traz tarefas semelhantes desenvolvidas anteriormente.

Mesmo com questões a serem melhoradas, como a assertividade do modelo e a usabilidade da ferramenta, percepções direcionam para benefícios da *ML Planning Poker* em tarefas realizadas há muito tempo ou que os membros não possuem experiência, trazendo maior segurança ao estimador no processo de estimativas. A proposta apresenta potencial pois valoriza o fator humano presente no *Planning Poker*, enquanto o AM apoia a tomada de decisão, melhorando o processo de estimativas.

Referências

Bloch, M., Blumberg, S., & Laartz, J. (2012). Delivering large-scale IT projects on time, on budget, and on value. *Harvard Business Review*, 5(1), 2-7.

Cohn, M. (2005). Agile estimating and planning. *Pearson Education*.

Dantas, E., Perkusich, M., Dilorenzo, E., Santos, D. F., Almeida, H., & Perkusich, A. (2018). Effort estimation in agile software development: an updated review. *International Journal of Software Engineering and Knowledge Engineering*, 28(11n12), 1811-1831.

El Emam, K., & Koru, A. G. (2008). A replicated survey of IT software project failures. *IEEE software*, 25(5), 84-90.

Idri, A., Hosni, M., & Abran, A. (2016). Systematic literature review of ensemble effort estimation. *Journal of Systems and Software*, 118, 151-175.

Tissot, A. A., Emer, M. C. F. P., & Bastos, L. C. Influence of the review of executed activities utilizing Planning Poker. In *2015 29th Brazilian Symposium on Software Engineering* (pp. 170-178). IEEE.

Tronto, I. F. de B., da Silva, J. D. S., & Sant'Anna, N. (2008). An investigation of artificial neural networks based prediction systems in software project management. *Journal of Systems and Software*, 81(3), 356-367.

Wen, J., Li, S., Lin, Z., Hu, Y., & Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology*, 54(1), 41-59.

Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA)*, 314-324.