LUDIFICAÇÃO NA ENGENHARIA DE SOFTWARE: APLICAÇÃO NO *AGILO FOR TRAC*

Trabalho de Conclusão de Curso

Engenharia da Computação

Rodrigo Cavalcanti Neto
Orientador: Prof. Joabe Bezerra de Jesus Júnior
Co-orientador: Prof. Gabriel Ramos Falconieri Freitas

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

RODRIGO CAVALCANTI NETO

LUDIFICAÇÃO NA ENGENHARIA DE SOFTWARE: APLICAÇÃO NO *AGILO FOR TRAC*

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2016.

ESCOLA POLITÉCNICA
DE PERNAMBUCO

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 4 de julho de 2016, às 9:00 horas, reuniu-se para deliberar a defesa da monografia de conclusão de curso do discente RODRIGO CAVALCANTI NETO, orientado pelo professor Joabe Bezerra de Jesus Júnior, sob título LUDIFICAÇÃO NA ENGENHARIA DE SOFTWARE, a banca composta pelos professores:

Eliane Maria Loiola

Joabe Bezerra de Jesus Júnior

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

☒ Aprovada ☐ Aprovada com Restrições* ☐ Reprovada

e foi-lhe atribuída nota: 9.0 (nove)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

ELIANE MARIA LOIOLA

JOABE BEZERRA DE JESUS JÚNIOR

* Este documento deverá ser encadernado juntamente com a monografia em versão final.

Dedico esse trabalho aos meus pais.

Agradecimentos

Primeiramente, agradeço minha família por ter guiado meu caminho, contribuir sempre o melhor para mim e pelo apoio durante minha caminhada.

Agradeço também aos meus orientadores que foram imprescindíveis para a realização do trabalho e pelas experiências que foram trocadas durante o processo de elaboração do mesmo.

Resumo

Desde algumas décadas atrás, a chamada crise de *software* começou a ser observada na atividade de desenvolvimento de *softwares*. À medida que essa crise evoluiu, vários problemas não previstos surgiram e comprometeram o processo de desenvolvimento em suas diferentes etapas de análise, projeto, construção, implantação ou manutenção, e consequentemente o produto final. Um fato que contribui ainda mais com o insucesso de projetos de *software* é a monotonia dos desenvolvedores e *stakeholders*, causada pela falta significativa de engajamento dentro da equipe e pela falta de oportunidade de utilizar seus talentos nas atividades do projeto. Para isso é proposta uma forma de promover o engajamento e motivação no desenvolvimento de *software*, mais especificamente aos profissionais que atuam na disciplina de codificação, através da ludificação. A ludificação é proposta através da aplicação de elementos de jogos na ferramenta de gerenciamento de tarefas Agilo *for* Trac com a finalidade de tornar o ambiente mais interativo e consequentemente aumentar a produtividade das equipes de desenvolvimento de software.

Palavras-chave: Ludificação, Agilo *for* Trac, Engajamento

Abstract

Since a few decades ago, the so-called software crisis began to be observed in the software development activity. As the crisis evolved, several unforeseen problems have emerged and committed the development process in its different stages of analysis, design, construction, implementation and maintenance, and therefore the final product. A fact that further contributes to the failure of software projects is the monotony of developers and stakeholders caused by significant lack of engagement within the team and the lack of opportunity to use their talents in project activities. Thus, it proposes a way to promote engagement and motivation in software development, specifically to professionals in coding discipline, through gamification. The gamification is proposed by applying game elements in the task management tool Agilo for Trac in order to make it more interactive environment and consequently increase the productivity of software development teams.

Keywords: Gamification, Agilo for Trac, Engagement

Sumário

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

Este capítulo inicia com a descrição do problema e a motivação para a realização deste trabalho. Em seguida, são relatados os principais objetivos a serem alcançados. Por fim, é detalhada a estrutura dos capítulos seguintes.

1.1 Motivação e Problema

Há alguns anos o desenvolvimento de *software* sofreu com o surgimento da crise de *software*, um problema que chamou a atenção dos profissionais da área de tecnologia da informação [1].

Segundo Rezende [2], a crise ocorre quando o sistema não satisfaz todos os envolvidos no projeto, por exemplo, clientes, desenvolvedores e usuários.

Com esse cenário a Engenharia de *Software* foi criada com o intuito de combater as causas que levaram a chamada crise de *software*.

À medida que a crise evoluiu, vários problemas não previstos surgiram e comprometeram o processo de desenvolvimento em suas diferentes etapas de análise, projeto, construção, implantação ou manutenção, e consequentemente o produto final. Logo, vários *frameworks* vêm sendo criados em busca da amenização ou do fim da calamidade crônica associada ao processo de *software*, assim rotulada a crise de *software* por Pressman [3].

Um fato que contribui ainda mais com o insucesso de projetos de *software* é a monotonia dos desenvolvedores e *stakeholders*, causado pela falta significativa de engajamento dentro da equipe e pela falta de oportunidade de utilizar seus talentos nas atividades do projeto [4]. Em busca da utilização de jogos para mudar a maneira como os negócios de vendas, *marketing*, pesquisa, desenvolvimento, produção e gestão operam Reeves e Read [5] acreditam que o uso de jogos redesenhará a forma de trabalhar, fazendo com que isso se torne cada vez mais semelhante a um jogo. A caracterização do ambiente de trabalho e dos processos deste, através de

jogos e de ludificação fazem com que o engajamento das equipes se torne cada vez mais frequente.

Dessa forma, é justificável a aplicação de ludificação dentro do processo de desenvolvimento de *software*, através da ferramenta Agilo *for* Trac para promover um maior engajamento dos desenvolvedores.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral dessa monografia é de promover a ludificação do processo de desenvolvimento de *software* ágil, através de uma ferramenta de gerenciamento de tarefas, buscando o aumento de engajamento dentro de equipes de desenvolvimento.

1.2.2 Objetivos Específicos

Os objetivos específicos desse trabalho são:

- Entender que elementos de jogos já são utilizados no processo de desenvolvimento de *software* e quais os resultados que elas promovem;
- Definir quais elementos de jogos serão utilizados no processo de ludificação para aumentar o engajamento das equipes de desenvolvimento de *software*;
- Definir quais métricas serão utilizadas para medir eficácia das equipes e habilidades dos desenvolvedores no processo de desenvolvimento de *software*;
- Projetar elementos de jogos e métricas definidos anteriormente, nas configurações do gerenciador de tarefas Agilo *for* Trac.

1.3 Estrutura da Monografia

O Capítulo 2 apresenta a fundamentação teórica, imprescindível para a compreensão do trabalho proposto, a saber: gerenciamento de atividades na engenharia de software e ludificação. O Capítulo 3 descreve a metodologia de ludificação utilizada para alcançar alguns objetivos específicos desta monografia, buscando entender trabalhos relacionados e propor um modelo para a utilização da ludificação no processo de desenvolvimento de *software*. No Capítulo 4, é apresentada a simulação de um estudo de caso exibindo o funcionamento dos elementos propostos na ferramenta de gerenciamento de tarefas Agilo *for* Trac. Por fim, o Capítulo 5 apresenta um resumo do trabalho final e discussões obtidas, tais como melhorias propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

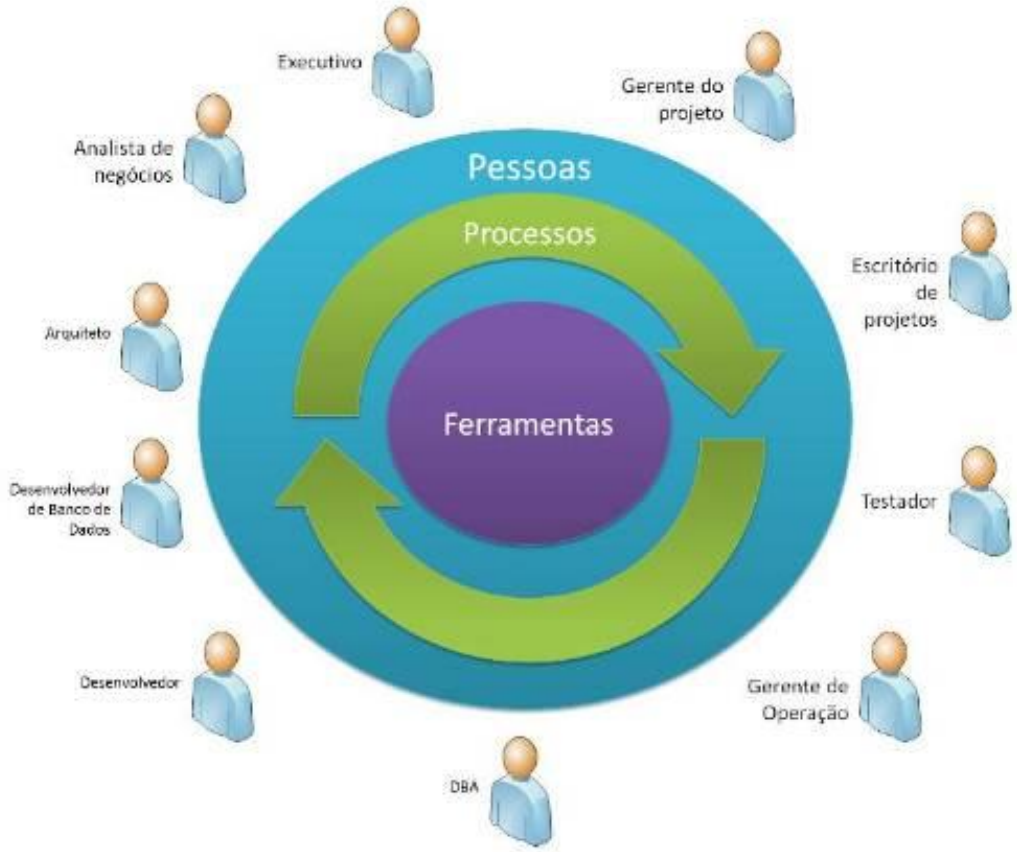
Este capítulo descreve brevemente o conteúdo teórico utilizado como base para tentar resolver o problema descrito no Capítulo 1. Inicialmente, a Seção 2.1 descreve sobre a ALM (*Application Lifecycle Management*) e a Engenharia de *Software*. A Seção 2.2 apresenta conceitos mais específicos da Engenharia de *Software* ágil e da utilização de ferramentas para apoio para a mesma. A Seção 2.3 aborda conceitos de ludificação a fim de saber como é definida.

2.1 ALM e Engenharia de *Software*

Com a incorporação de TI (Tecnologia da Informação) no dia-a-dia das empresas, algumas aplicações tornaram-se ferramentas para auxiliar o processo de produção de produtos e serviços. Considerando que as aplicações de uma empresa são bens que não possuem um corpo físico e são resultantes de transações econômicas, portanto essas necessitam de cuidados na aquisição, construção, utilização e na manutenção. Dentro desse contexto Condé [6] define ALM como o processo que guia a vida útil de uma aplicação. Esse processo não apenas observa qual é o método de construção, mas também se preocupa como a empresa gerencia este bem, ou seja, trata-se da união da Engenharia de Software com a área de gestão de negócios.

O processo do ALM é estruturado em cima de três pilares que quando unidos, fornecem os recursos necessários para que as empresas possam gerenciar o ciclo de vida das suas aplicações. Esses três pilares se complementam e são determinados como: pessoas, processos e ferramentas. Pode-se verificar na Figura 1 uma representação da estrutura do ALM.

Figura 1. Representação da estrutura do ALM.



Fonte: Condé [6].

Na Engenharia de *Software* existe o conceito de processo de *software* que segundo Pressman [3] é uma metodologia para as atividades, ações e tarefas e define uma abordagem adotada para desenvolver um *software* de alta qualidade.

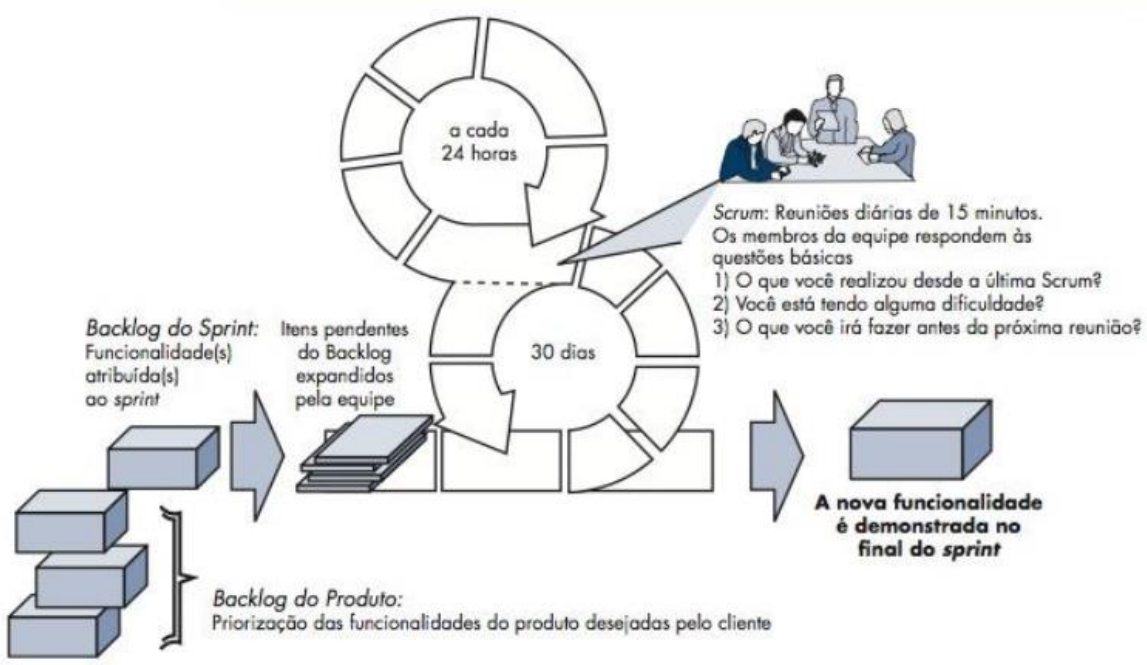
As metodologias consideradas tradicionais têm como característica marcante serem divididas em etapas ou fases. Essas fases são muito bem definidas e englobam atividades como Análise, Modelagem, Desenvolvimento e Testes. Um marco é gerado na conclusão de alguma das fases referidas, o que geralmente é algum documento, protótipo do *software* ou mesmo uma versão do sistema. Muitas dessas metodologias tradicionais se utilizam do modelo em cascata o que dificulta o controle do projeto, pois a cada alteração em determinado ponto do projeto, como os requisitos, será necessário uma volta ao início do mesmo para alteração de documentação ou outro marco [3].

2.2 Engenharia de *Software* Ágil

Com a intenção de sanar os problemas da Engenharia de *Software* tradicional, surgem os métodos ágeis, que segundo Pressman [3] oferece benefícios importantes, porém o desenvolvimento ágil não é indicado para todos os projetos, produtos, pessoas e situações. Para isso a agilidade incentiva à estruturação e as atitudes em equipe facilitando a comunicação entre os membros, além de enfatizar a entrega rápida do *software* operacional através da desvalorização de artefatos intermediários.

Um das metodologias que propõe a agilidade comentada é o *framework Scrum* que utiliza o modelo iterativo e incremental para gerenciamento e desenvolvimento das suas atividades de processo. O fluxo geral do *Scrum* pode ser visualizado na Figura 2.

Figura 2. Fluxo geral do processo *Scrum.*



Fonte: Pressman [3].

Segundo Noyes [7] o Scrum coloca uma forte ênfase na comunicação, trabalho em equipe e flexibilidade. Este método define um conjunto de ações de desenvolvimento e de conceitos tais como o *backlog,* a *sprint*, e as reuniões diárias.

O *backlog* é o local onde se registra todo o trabalho pendente (requisitos ou funcionalidades) organizado por prioridades. Ressalta-se que novas funcionalidades podem ser adicionadas a ele a qualquer momento, porém, o gerente do produto deve avaliar esta funcionalidade e atualizar as prioridades.

Outra prática do *Scrum* é a definição de uma *sprint*, determinada pelas divisões das funcionalidades do *backlog* e esta deve ser atendida num curto prazo de tempo, no máximo 30 dias e nela não deve haver alterações, quaisquer modificações devem ser aplicadas nas *sprints* seguintes. Os *backlogs* pertencentes à *sprint* são chamados de *backlogs* da *sprint*. É na *sprint* que o trabalho de desenvolvimento é realizado para entregar o mais rápido possível um incremento funcional do *software* ao cliente.

Durante as *Sprints* é necessário realizar as Reuniões diárias que se caracterizam por reuniões tipicamente curtas (15 minutos) realizadas diariamente pela equipe. Nelas são levantadas três questões pontuais para todos os membros da equipe listados na Tabela 1, com a finalidade de entender o que cada um fez no dia anterior, se tem alguma dificuldade para realizar o trabalho atual e o que pretendem fazer no dia.

Tabela 1. Tabela de perguntas realizadas nas reuniões diárias do *Scrum.*

O que você realizou desde a última reunião da equipe?
Quais obstáculos está encontrando?
O que planeja realizar até a próxima reunião da equipe?

No *Scrum* também temos um líder de equipe que é chamado de *Scrum Master*. Ele é responsável por conduzir a reunião diária e avaliar as respostas dos integrantes. O objetivo do *Scrum Master* também é remover obstáculos.

2.2.1 Ferramentas de apoio ao *Scrum*

Com o avanço da tecnologia foram criadas ferramentas que dão suporte a implementação do *Scrum* e o mercado as disponibiliza de uma forma variada que vão desde planilhas automatizadas, a pacotes de softwares *open-source* até mesmo ferramentas proprietárias dedicadas à gestão e suporte aos eventos e atividades do

Scrum. Na Tabela 2 é possível visualizar algumas ferramentas disponíveis no mercado para apoio ao *Scrum* listadas por Pereira [8].

Tabela 2. Tabela de Ferramentas de apoio à implementação do *Scrum.*

Nome	Tipo da Licença	Considerações
Scrummy/ Scrummy Pro	*FREE*/ Monthly Payment	• Ferramenta *web-based* hospedada em web site proprietário, com os dados do projeto em nuvem; • Sem necessidade de instalação e (ou) configuração; • Suporte a alguns eventos *Scrum* como: i) criação de *user stories* e ii) acompanhamento das *user stories* por *taskboard*; • Na versão Pro, agrega suporte aos eventos: i) criação de múltiplas *sprints*; ii) geração e coordenação de múltiplos *backlogs* (*Product* e *Sprint*); iii) Acompanhamento de andamento de projetos por *Burndown Charts*.
ScrumWorks Basic/ ScrumWorks Pro	*FREE*/ *Pay Per User/Per Year*	• Ferramenta WinForms, necessário instalação e configuração; • Composição de dois (2) programas, cliente e servidor. • Ferramenta servidor, usada pelo *Scrum Master,* onde são gerenciados os eventos de planejamento; • Ferramenta Cliente, de uso da Equipe de Desenvolvimento, com o acompanhamento de tarefas, *dashboards*, *tickets* e etc; • Na versão Pro, é possível implementar todos os eventos recomendados do *Scrum*.
Pango Scrum	*FREE* (BETA)	• Ferramenta *web-based* hospedada em *web site* proprietário, com os dados do projeto em nuvem; • Sem necessidade de instalação e (ou) configuração; • Suporte aos eventos do *Scrum*: i) gerenciamento do *Product Backlog*; ii) Planejamento de *Spints*; iii) Agendamento de Eventos com Reuniões, *reviews* e

Nome	Tipo da Licença	Considerações
		retrospectivas; • *Dashboards* e gráficos de acompanhamento.
iceScrum	*FREE (Open Source)* *iceScrum* PRO	• Ferramenta *web-based* disponível como *open source;* • Ferramenta completa, com suporte as funcionalidades de *Product e Sprint Backlog*, *planning poker*, *dashboards*, e suporte a gestão de eventos como Reuniões de Planejamento; • Fabricante oferece a versão PRO com privilégios de hospedagem do projeto *in cloud*.
Scrum Half	*FREE* *Pay per User*	• Ainda na versão BETA, ferramenta brasileira *web-based* hospedada em *web site* proprietário, com os dados do projeto em nuvem; • Gratuita para o uso de até três (3) pessoas (pague somente pelo excedente); • Suporte *online* para ajuda e suporte a erros; • Funcionalidades como: i) Quadro de Tarefas; ii) *Product Backlog* iii) Gráficos de acompanhamento, *Burndown* e relatórios; iv) Gestão de *releases* e *sprints*; v) *Feed* de notícias do projeto e *Online Chat* entre os integrantes.
Mingle – *Agile Project Management*	*Pay per User*	• Considerada na comunidade ágil como uma das melhores ferramentas ágeis; • Criada pela empresa ThoughtWorks, cujo um dos especialistas é Martin Fowler referência em desenvolvimento ágil de software; • Suporte a todos os eventos e atividades do *Scrum*; • *Templates* e suporte a *Kanban*; • Customização de cartões, quadros e *user stories;* • Acompanhamento e gerenciamento de projeção e progresso do projeto por gráficos,

Nome	Tipo da Licença	Considerações
		burndown e relatórios customizados.
Agilo for Trac (older Agilo for Scrum)	*FREE* *Pay per User*	• Ferramenta *Web-Based*, necessário servidor de projetos, montagem e configuração de ambiente; • Possui integração a ferramenta TRAC; • Suporte aos eventos *Scrum* e orientada a um fluxo de trabalho intuitivo por função, ou seja, responsabilidade *Scrum* ligada a funcionalidade no *Agilo*.

2.2.2 Agilo For Trac

O Agilo *for* Trac é uma ferramenta *web-based* para suporte e utilização do *framework Scrum*. Trata-se de uma ferramenta flexível e com grande quantidade de funcionalidades configuráveis para fluxos de trabalhos específicos [9].

Em relação às funcionalidades como pode ser observado na Tabela 3, o Agilo *for* Trac se utiliza dos conceitos do *Scrum*, com a existência de *tickets* representando requisitos, histórias e tarefas que podem possuir *links* uns para os outros, aplicando rastreabilidade no ambiente de gerenciamento de atividades.

Tabela 3. Tabela de funcionalidade do Agilo *for* Trac (comparativo entre versões).

Funcionalidades	Agilo (Versão gratuita)	Agilo (Versão Pro)
Suporte completo para todos os três papéis no *Scrum*: Desenvolvedor, *Scrum Master* e *Product Owner*, com diferentes funcionalidades para dar suporte a seus fluxos de trabalho.	X	X
Tickets hierárquicos para criar *link* de tarefas para histórias (e para requisitos se necessário)	X	X

Funcionalidades	Agilo (Versão gratuita)	Agilo (Versão Pro)
Rastreabilidade completa de requisitos através de histórias e tarefas para o *commit* que entrega a funcionalidade.	X	X
Integração com o Subversion: tarefa, defeito etc. podem ser fechados com *commits* do Subversion.	X	X
Forte integração com o seu código fonte: código de navegação, *link* para alterar conjuntos e *diffs* de exibição com diferentes sistemas de controle de versão (Subversion, Mercurial, Git, Bazaar and Perforce)	X	X
Exportação para Excel (CSV) e importação com edição em massa e exclusão em massa.	X	X
Product backlog priorizado e suporte para mapeamento de histórias para *sprints*	X	X
Suporte a múltiplos projetos e múltiplos times	X	X
Múltiplos *backlogs*: *backlog* de defeito, *backlog* de impedimento	X	X
Suporte a *Burndown chart* com equipes distribuídas em vários fusos horários (suporte a fusos horários)	X	X
Suporte para contingentes, para limitar as tarefas imprevisíveis como pedidos de suporte ou correções de defeitos para uma caixa de tempo	X	X
Completamente configurável: adição de tipos e campos customizados.	X	X
Link de histórias a múltiplos requisitos para melhorar a rastreabilidade	X	X
Wiki integrado	X	X
Extensível com plug-ins gratuitos	X	X

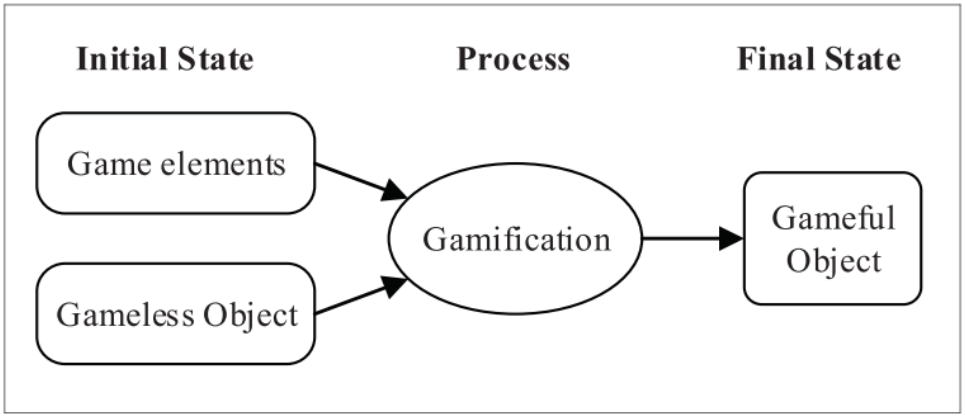
Funcionalidades	**Agilo (Versão gratuita)**	**Agilo (Versão Pro)**
Gestão integrada da equipe: Ao digitar a disponibilidade da equipe o agilo faz o cálculo.	X	X
Edição de atributos importantes da história/tarefa diretamente na visão do *backlog*	X	X
Drag&drop considera hierarquias	X	X
Drag&drop inteligente: priorizar seu *backlog* com *drag&drop*	X	X
Apoio profissional incluindo SLA		X
Whiteboard integrado		X
Exportação e exibição de *whiteboard*		X
Criação de novos *tickets* diretamente da visão de *backlog.*		X
Edição de todos os atributos de histórias/tarefas do *backlog* sem recarga		X
Visões de configurações são armazenadas para cada usuário		X
Suporte a *kanban* através de colunas de *whiteboard* configuráveis		X
Serviço hospedado disponível		X

2.3 Ludificação

A popularidade da ludificação vem crescendo nos últimos anos, baseada na pesquisa de Yohannis e outros [10] pelo termo *gamification*, cuja palavra no português é representada pelo termo ludificação. O Google Trend mostrou que pessoas já pesquisavam por este termo desde o ano de 2010 e que, com o passar dos anos, a tendência de pesquisa de *gamification* teve um crescimento contínuo, indicando também o crescimento da popularidade da ludificação.

Na literatura existem dois conceitos que podem ser confundidos dentro do contexto atual, são eles a ludificação e o conceito de jogos. Para Yohannis et al [10] a definição de um objeto com jogo se caracteriza pela identificação de nove características: jogador, ambiente, regra, desafio, interação, meta, experiência emocional, resultados quantificáveis e consequências negociáveis. A presença de mais características vai resultar em um objeto com uma maior identidade de jogo, enquanto na ausência de todas elas o objeto não é um jogo. Do outro lado, Yohannis e outros [10] ainda definem ludificação, porém para isso são usadas diferentes abordagens, uma pelo significado léxico do termo, e a segunda através do ponto de vista de processo. A partir do significado léxico, pode-se inferir ludificação como a ação de tornar algo em um jogo. Essa ação também pode ser determinada como um processo, onde um objeto com poucas características de jogo (estado inicial) é ludificado (ação), e como produto, tem-se um objeto com mais identidade de jogo (estado final), portanto, houve o processo de ludificação como apresenta a Figura 3.

Figura 3. Uma definição de ludificação do ponto de vista de processo.



Fonte: Yohannis [10].

Dado que o processo de ludificação pode resultar em um objeto mais ludificado, logo o excesso da ludificação pode causar a condição de *overgamification,* que é explicada por Yohannis et al [10] pelo fato da porcentagem da característica de jogo superar a porcentagem da característica básica do objeto, podendo levar a inabilidade desse objeto de executar sua função essencial.

Portanto, Yohannis e colaboradores [10], através de uma comparação de definições de ludificação, assumem que existem alguns itens que compõem a definição da mesma, que são **a caracterização de um processo de integração, elementos de jogos, objetos com falta de característica de jogos, objetivo imediato, objetivos posteriores** e serão descritos a seguir.

A caracterização de um processo de integração: ludificação é um processo que possui estado inicial e final de um objeto e nele é realizada uma ação de forma que anteriormente no estado inicial ele possui poucas características de jogos, ou nenhuma característica e no seu estado final possui a adição de certas características de jogos.

Elementos de jogos: a ludificação se utiliza de elementos de jogos.

Objetos com falta de característica de jogos: no processo, o estado inicial do objeto deve ter falta de características de jogos e ter ao menos uma característica essencial, que não seja uma das características de jogos.

Objetivo imediato: deve possui como objetivo imediato aumentar o grau ou a quantidade de características de jogo em um objeto.

Objetivos posteriores: após o objetivo imediato ser atingido, outra meta é definida em relação ao contexto do objeto.

Os elementos de jogos são definidos por Zichermann e Cunningham [11] como uma série de ferramentas que prometem obter uma resposta significativa a partir dos jogadores. Nesse contexto são definidos como elementos primários os mecanismos de pontuação, níveis, quadros de liderança, emblemas/distintivos, desafios/missões, *onboarding* e *engagement loops,* que serão brevemente definidos a seguir.

Pontuação – Os jogadores recebem uma recompensa em forma de pontos quando completam determinada atividade. A partir do somatório de pontos é possível estabelecer um sistema de pontuação para os jogadores.

Níveis – Os usuários estão em um nível que aumenta à medida que atingem um determinado número de pontos.

Quadros de liderança – um *ranking* com os melhores jogadores é apresentado a todos os jogadores para aumentar a competitividade. A posição no *ranking* pode ser definida por pontos, níveis ou quantidades de emblemas, por exemplo.

Emblemas/distintivos – eles representam certas realizações do usuário. A Figura 4 apresenta uma exemplo de emblemas de diferentes realizações na ferramenta RedCritter Tracker.

Figura 4. Exemplo de emblemas do RedCritter Tracker.

Fonte: https://www.redcrittertracker.com/.

Desafios/missões – as tarefas que o jogador precisa completar são apresentadas como uma meta, com elementos de jogo adicionais como uma narrativa, o que as torna mais atraente.

***Onboarding* –** o mecanismo de *onboarding* se caracteriza pela integração de um novo jogador ao ambiente, onde ele é sempre guiado para o caminho do sucesso a fim de que a complexidade no jogo seja apresentada aos poucos. A Figura 5 mostra Frontier Jack, uma seta amarela, e um pedaço de grama do jogo FrontierVille. Tudo o que um jogador tem que fazer é clicar logo abaixo da seta amarela.

Engagement loops – O *engagement loops* ilustrado na Figura 6, busca além do engajamento dos desenvolvedores o reengajamento dos mesmos através da atribuição de recompensas para que estes se mantenham sempre motivados.

Figura 5. Processo de *onboarding* do Frontier Ville's.

Fonte: Zichermann e Cunningham [11].

O processo de *onboarding* do Frontier Ville's ilustrado na Figura 5, é simples, lentamente revela a complexidade e não dá opção para falhas.

Figura 6. Exemplo de *engagement loop* social.

Fonte: Zichermann e Cunningham [11].

O *engagement loop* social apresentado na Figura 6, foi projetado para maximizar o engajamento e reengajamento do jogador.

Capítulo 3

Metodologia de Ludificação

Este capítulo detalha a proposta de metodologia utilizada para promover ludificação em uma ferramenta de controle de tarefas onde serão utilizadas para o estudo as tarefas de codificação de acordo com as teorias apresentadas no capítulo anterior.

3.1 Trabalhos Relacionados

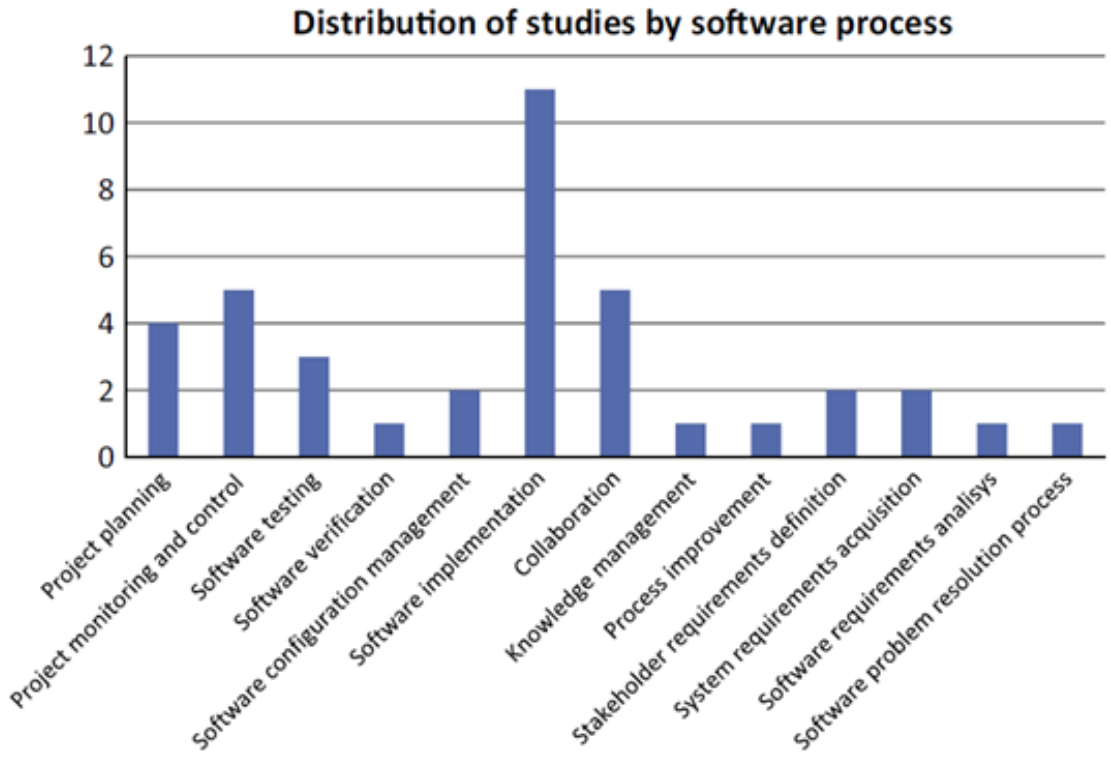
Através de um mapeamento sistemático da literatura Pedreira e outros [12] classificaram a distribuição de estudos primários onde existem objetos de ludificação, pelas diferentes disciplinas da Engenharia de *Software*. O resultado do mapeamento mostrou que o processo de desenvolvimento de *software* no contexto de codificação, tem uma maior quantidade de estudos primários como pode ser observado na Figura 7. Isso não é surpreendente dado que o processo de desenvolvimento de *software* contém algumas características que fazem dele adequado para aplicação de ludificação.

Ainda em [12] é possível identificar quais elementos de ludificação são mais utilizados dentre todas as disciplinas da Engenharia de *Software* consideradas na pesquisa, dado que alguns estudos possuem a natureza teórica enquanto que outros possuem a aplicação em atividades reais. Considerando que os estudos primários abordam alguns elementos com termos diferentes é possível visualizar na Figura 8 que a utilização de pontuação parece ser o mecanismo mais utilizado seguindo por *badges,* que se caracteriza pela atribuição de distintivos ou emblemas, e na terceira colocação a utilização de sistema de votação.

A utilização de elementos de ludificação dentro do processo de desenvolvimento de *software* requer uma ou mais métricas para se medir alguns fatores do processo. Na literatura os autores se utilizam de métricas diversas, como é o caso de Melo e colaboradores em [13] que utilizam uma métrica de obtenção de pontos dos desenvolvedores que é calculada a partir da complexidade do código

produzido por eles. Para isto foram utilizados mecanismos básicos de pontuação e de *ranking* com a finalidade de definir os desenvolvedores que ganhariam recompensas ao realizar a codificação de uma funcionalidade.

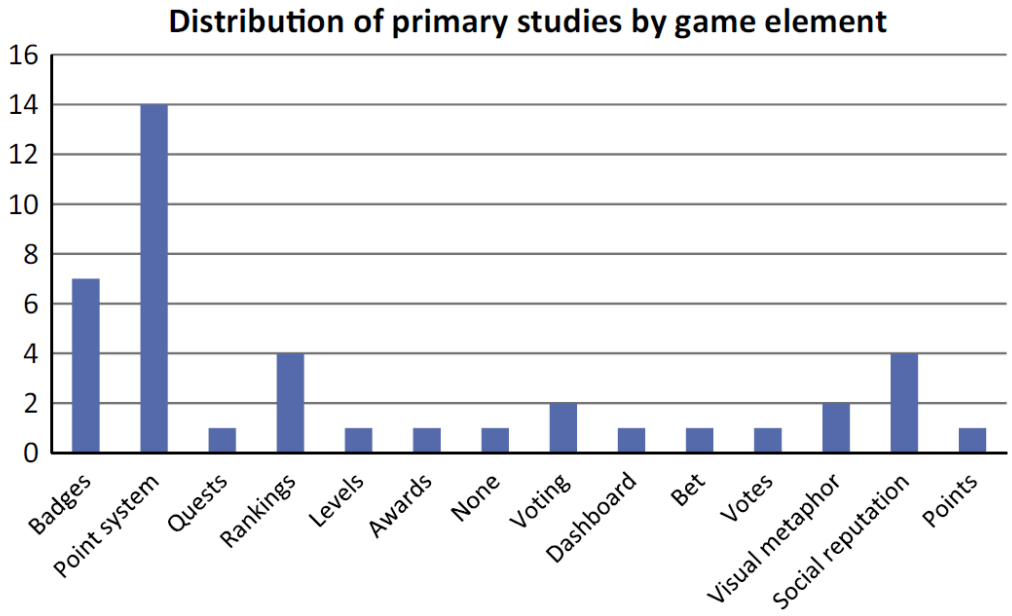
Figura 7. Distribuição de estudos primários por área de processo.



Fonte: Pedreira [12].

Passos [14] propõe a incorporação de alguns elementos presentes em jogos na estrutura de uma ferramenta de gerenciamento de tarefas buscando um aumento da diversão obtida na realização das tarefas. Passos [14] ainda em seu trabalho, discute um conceito de ludificação diferente do conceito apresentado no presente trabalho, porém a ação de aplicar elementos de jogos em atividades cotidianas é comum entre ambos. A principal diferença do trabalho de Passos [14] para o atual é que o primeiro apresenta uma proposta de ludificação de uma ferramenta de gerenciamentos de tarefa para a Engenharia de *Software,* enquanto que o atual apresenta um modelo de ludificação com métricas e elementos de jogos bem definidos na ferramenta específica Agilo *for* Trac para o processo de codificação de *software*.

Figura 8. Distribuição de estudos primários por elemento da ludificação.

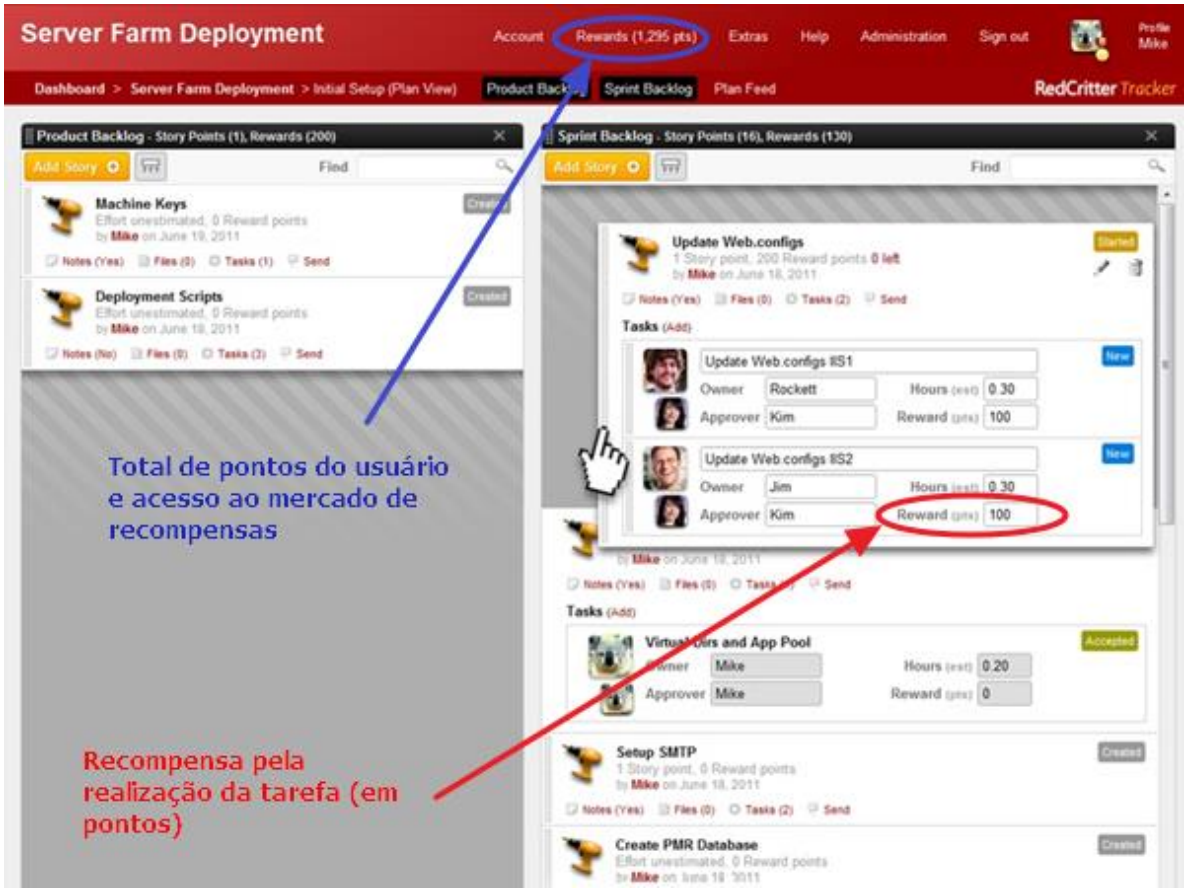


Fonte: Pedreira [12].

Um ferramenta que tem uma proposta mais próxima com a abordagem do trabalho atual é ferramenta de gerenciamento de projetos RedCritter Tracker [15], uma ferramenta de apoio à gestão do projeto que a aplica a ludificação a times ou equipes de desenvolvimento de *software* com alguns elementos de jogos como visibilidade de conjuntos de habilidades dos empregados, recompensas personalizáveis e emblemas, a Figura 9 apresenta a tela de tarefas dos *backlogs* da ferramenta. O RedCritter Tracker oferece a possibilidade de personalização de habilidades e de escolhas de recompensas através de um mercado de recompensas ilustrado na Figura 10, onde a moeda de troca são os pontos acumulados pelos empregados durante o processo de desenvolvimento. Porém as métricas aplicadas e as regras não seguem a mesma ideia do presente trabalho.

A próxima seção faz menção aos elementos de jogos usados para o processo de ludificação proposto no desenvolvimento de *software* através de uma ferramenta de gerenciamento de tarefas.

Figura 9. Tela de planejamento do RedCritter Tracker.



Fonte: https://gigaom.com/2011/07/22/redcritter-tracker-agile-project-management-with-gamification/.

Figura 10. Exemplo do mercado de recompensas do RedCritter Tracker.

Fonte: https://www.redcrittertracker.com/home.aspx.

3.2 Modelo Proposto para Ludificação

Para a aplicação da ludificação na ferramenta de gerenciamento de tarefas, definiu-se que os elementos de jogos a serem usados são: sistema de pontuação, *ranking* de pontos, distintivos ou emblemas esses com o propósito de ofertar o *feedback* do desenvolvimento de *software*, e a proposta de recompensas e punições com a finalidade de promover a experiência emocional de engajamento e motivação aos desenvolvedores. A definição dos elementos de jogos e de como estes elementos serão utilizados foi realizada a partir da experiência em projetos de uma empresa multinacional, que por motivo de sigilo não pode ter seu nome divulgado. O estudo de trabalhos relacionados também auxiliou nas definições realizadas.

3.2.1 Sistema de Pontuação

Na aplicação do mecanismo de sistema de pontuação, cada desenvolvedor acumula pontos conforme realiza suas tarefas de codificação. Assume-se que o sistema de pontuação possibilita mensurar a produtividade de todos os desenvolvedores e equipes em cada um dos projetos onde participaram e na organização, considerando o somatório de pontos de todos os projetos onde houve a aplicação do modelo de ludificação.

Cada tarefa possui duas variáveis fundamentais para o sistema de pontuação, a primeira é a variável de tempo estimado da *baseline* (***bl_remaining_time***). A *baseline* é definida por Pressman [3] como um marco de referência no desenvolvimento de um software. A segunda variável é o total de esforço gasto na realização da tarefa em horas (***total_hours***). A variável ***bl_remaining_time*** representa em horas, a quantidade de unidades de medida de *software* que são gastas para cada tarefa de codificação, este valor é baseado em uma base histórica de estimativas. Através da codificação de casos de usos de projetos anteriores é possível obter uma média de tempo gasto para cada unidade de ponto de função, estas informações formam a base histórica de estimativas.

Dado que uma empresa possui base histórica de estimativas em determinada medida de *software,* é possível determinar o número de horas necessárias para codificar uma unidade dessa medida, com base nisso é possível determinar a

estimativa de tempo gasto na codificação de um novo caso de uso, essa estimativa é representada pela variável ***bl_remaining_time.*** .

Com base no tempo estimado *da* baseline (***bl_remaining_time***) e no total de esforço gasto na realização da tarefa de codificação em horas (***total_hours***), pode-se calcular outro fator, nomeado neste trabalho como produtividade aparente, que significa a produtividade que o desenvolvedor aparentemente possui na codificação da funcionalidade, antes da mesma passar pela fase de testes.

O cálculo da produtividade aparente para cada tarefa é equivalente à razão da ***bl_remaining_time*** pelo ***total_hours***, assim, quanto mais esforço em horas um desenvolvedor gasta em uma tarefa, menor é sua produtividade aparente. O nível de normalidade considerado quando o ***total_hours*** é exatamente o mesmo da ***bl_remaining_time***, resultando em uma produtividade aparente igual a cem por cento (100%). Quanto mais o desenvolvedor gasta horas e ultrapassa a estimativa, menor é sua produtividade aparente.

Dado que se pode calcular a produtividade aparente, esta é considerada como o fator de pontuação de cada tarefa de codificação, para isso é utilizada a própria ***bl_remaining_time*** como pontuação base para cada tarefa. Assim, se uma dada tarefa possui oitenta horas como estimativa, então, ela possui um valor agregado de oitenta pontos, e este é multiplicado pelo fator de pontuação para determinar quantos pontos o desenvolvedor ganhará ao terminar a codificação de uma funcionalidade. A equação (1) representa a quantidade de pontos ganhos por cada tarefa de codificação realizada por desenvolvedor.

$$Pontos = bl\_remaining\_time \times \frac{bl\_remaining\_time}{total\_hours} \quad (1)$$

3.2.2 Ranking de Pontos

Cada tarefa realizada atribui uma quantidade de pontos ao dono da mesma, e o somatório delas atribui a cada desenvolvedor uma possibilidade de ser ranqueado.

O mecanismo de *ranking* possibilita a competitividade entre os desenvolvedores. A exibição de posicionamento de todos é realizada pelo sistema

de pontuação. Assume-se que os mais produtivos possuem melhor colocação no *ranking*.

São determinadas três categorias de *ranking*, o conjunto dos mais produtivos da organização, de cada projeto e de cada entrega. O conjunto dos mais produtivos da organização utiliza todos os pontos obtidos pelo desenvolvedor desde a aplicação da ludificação na organização. O *ranking* dos mais produtivos de cada projeto será composto apenas pelos desenvolvedores que participam do determinado projeto, este *ranking* se perdura até o fim do mesmo. E por último, os mais produtivos de cada entrega é o *ranking* onde é destacada a produtividade dos desenvolvedores dentro de uma *milestone*, que se caracteriza pelos marcos de entregas parciais do projeto.

3.2.3 Distintivos/Emblemas

Outra forma de destacar os desenvolvedores é a atribuição de distintivos por realizações, às vezes positivas e às vezes negativas. Este mecanismo possui duas funções diferentes no módulo proposto, a primeira é de destacar diferentes habilidades fazendo com que não haja necessariamente um melhor desenvolvedor, mas sim, bons desenvolvedores com diferentes qualidades. A segunda função é de destacar quem possuiu dificuldades em determinado projeto ou *milestone*, esse destaque tem a função de mostrar à equipe quais as dificuldades do desenvolvedor, podendo ser associadas às tecnologias para um determinado projeto, ou até mesmo a uma determinada funcionalidade. Os distintivos são atribuídos em fases diferentes do projeto. Foram definidos três tipos de distintivos diferentes, são eles: **estimador de esforços mais preciso**, **desperdiçador de esforços** e **fidelidade de registro de horas**.

Considera-se que o distintivo do **estimador de esforços mais preciso** tem a característica de destacar o desenvolvedor que possui maior habilidade de estimar suas atividades, ou seja, quanto mais precisa for sua estimativa de esforço em horas (***remaining_time***) em relação ***total_hours*** na atividade, melhor estimador será o desenvolvedor. A variável *remaining_time* representa a estimativa de esforço realizada pelo desenvolvedor de acordo com sua experiência em projetos anteriores, essa estimativa é realizada antes de iniciar a atividade de codificação do caso de uso e não há acesso a estimativa da *baseline* (*bl_remaining_time*) para realizar este

processo. O cálculo de precisão de estimativa de determinada tarefa é dado pela razão das horas efetivamente gastas pelo desenvolvedor nela (***total_hours***) pelas horas estimadas pelo mesmo (***remaining_time***), sendo considerado como fator de estimação. Portanto, como cada desenvolvedor pode possuir uma quantidade de tarefas, então, assume-se que o melhor estimador será aquele que em média possuir um fator de estimação mais próximo do valor 1,00. A média de estimação é calculada como pela razão do somatório dos módulos dos fatores de estimação de cada atividade de um desenvolvedor pela quantidade dessas.

Com o intuito de mostrar as dificuldades de um desenvolvedor e incentivar que os demais sejam mais atentos, o distintivo do **desperdiçador de esforços** apresenta aquele que gasta proporcionalmente mais tempo corrigindo defeitos relativos às suas tarefas de codificação que os demais desenvolvedores. Este distintivo baseia-se na média da relação da quantidade de esforço gasto para correção de defeito se do tempo estimado da *baseline (**bl_remaining_time**)*. Este distintivo é atribuído aos desenvolvedores por *milestone* e por projeto.

Em algumas situações o desenvolvedor gasta pouco tempo para a realização de uma tarefa que possui um alto esforço estimado da *baseline (**bl_remaining_time**).* Isso pode significar um falso *status* de produtivo, pois a quantidade de esforço gasto na correção de defeitos desta tarefa pode ser tão alta quanto o tempo economizado na codificação. Na tentativa de evitar o falso produtivo, considerado nesse trabalho o desenvolvedor que possui produtividade aparente mais elevada que a produtividade real, aplicou-se uma penalidade na pontuação do desenvolvedor pela quantidade de esforço gasto na correção de defeitos. Para cada defeito de um caso de uso codificado, é realizado um decréscimo de pontos da quantidade de esforço gasto para a correção dos defeitos.

Outro emblema caracterizado neste trabalho é o de **fidelidade de registro de horas**. Este emblema é uma forma de medir o esforço registrado para cada desenvolvedor ou para cada equipe.

Cada atividade realizada pelos desenvolvedores deve possuir o registro diário de esforços registrados no formato de horas, o registro de esforços dá suporte ao gerente de projeto, ao próprio desenvolvedor e a equipe para que realizem o acompanhamento do andamento de determinada tarefa.

Para cada dia da semana que o desenvolvedor registra seu esforço de maneira correta, considerando a alocação deste, ele terá uma promoção de status, e no caso oposto onde o desenvolvedor ao fim do dia não registra o total de esforços gastos para suas atividades, terá um rebaixamento de status. O *status* do desenvolvedor reflete o quanto ele é fiel às suas atividades e cuidadoso com seu tempo. Dentro do contexto cada desenvolvedor possui um nível de representação de fidelidade como exibido na Tabela 4, ao registrar os esforços do dia ele ganha um ponto que é utilizado para promoção de categoria e ao deixar de registrar ele perde um ponto e consequentemente é rebaixado de categoria.

Tabela 4. Tabela de fidelidade de esforço gasto.

NÍVEL	DESCRIÇÃO
PLATINUM	**Maior que 4 até 5 pontos**
DIAMANTE	**Maior que 3 até 4 pontos**
OURO	**Maior que 2 até 3 pontos**
PRATA	**Maior que 1 até 2 pontos**
BRONZE	**Maior que 0 até 1 ponto**
NEUTRO	**0 ponto**
LATÃO	**Menor que 0 até -1 ponto**
MADEIRA	**Menor que -1 até -2 pontos**
PLASTICO	**Menor que -2 até -3 pontos**
PAPEL	**Menor que -3 até -4 pontos**
ÁGUA	**Menor que -4 até -5 pontos**

Assim, para cada semana o desenvolvedor terá uma classificação de acordo com o esforço registrado. A Cada semana o desenvolvedor terá a possibilidade de possuir um *status* diferente e no fim dela o *status* geral do desenvolvedor é atualizado, logo, cada um deles possuirá dentro da organização um *status* que poderá ser atualizado a cada semana.

3.2.4 Recompensas e Punições

Como proposta do trabalho, a atribuição de recompensas e/ou punições promove a possibilidade de aumentar o engajamento e a motivação dentro das equipes de desenvolvimento. A atribuição de recompensas ocorre quando um desenvolvedor realiza uma conquista ou uma série delas, de modo a beneficiar o ambiente em que se encontra, seja ele o andamento do projeto, a conclusão do mesmo ou quaisquer outras conquistas que contribuam para a evolução da equipe. Enquanto que a atribuição de punições é realizada aos desenvolvedores que tenham fracasso em alguma de suas atividades. Ambas as atribuições podem ser realizadas de diferentes maneiras e em diferentes fases do projeto, dependendo de como é aplicado esse modelo de ludificação.

Capítulo 4

Estudo de Caso

Este capítulo mostra um estudo de caso que usa a ferramenta de gerenciamento de atividades Agilo *for* Trac através da aplicação da metodologia detalha no Capítulo 3.

4.1 Configurações do Ambiente

Considera-se uma organização que trabalha em um projeto de desenvolvimento de um *software* nomeado sistema Alfa. A organização utiliza uma metodologia ágil de desenvolvimento de *software* e o sistema de controle de tarefas Agilo *for* Trac que será referido no presente trabalho apenas como Agilo. Como técnica para a medição de projetos a organização se utiliza de Pontos de Função (PF), que segundo Vazquez [16], é uma unidade de medida de funcionalidades fornecidas por um *software* do ponto de vista do seu usuário.

Para a implementação do módulo que aplica ludificação ao Agilo, se faz necessário que a organização em questão configure o seu ambiente Agilo.

4.1.1 Configuração do Agilo

O Agilo é utilizado com ao menos três tipos de tíquetes diferentes para atribuir tarefas de codificação de casos de uso e correção de erros aos desenvolvedores, como a Tabela 5 apresenta.

Tabela 5. Tabela de tipos de tíquetes.

Tipo do Tíquete	Descrição
Tarefa	Deve-se criar uma tarefa que representa o caso de uso.
Sub Tarefa	O caso de uso deve possuir sub-tarefas que representam as fases de implementação do mesmo, por exemplo, sub-tarefas de codificação ou de testes.

Tipo do Tíquete	Descrição
Defeito	Para cada defeito na codificação do caso de uso deve-se criar um tíquete de defeito.

Nos tíquetes de Sub Tarefa e Defeito devem existir três atributos, para que os cálculos dos elementos de jogo possam ser realizados, os atributos são:

- O total de esforço gasto no determinado tíquete em horas (***total_hours***);
- Estimativa de esforço gasto da *baseline* em horas, que é calculada por uma base histórica de estimativas e pela quantidade de PFs do caso de uso, onde cada unidade de ponto de função pode ser traduzida para uma quantidade de horas (***bl_remaining_time***);
- Estimativa de esforço gasto do desenvolvedor em horas (***remaining_time***);
- Adição de esforço gasto no tíquete em horas (***hours***).

4.2 Cenário de Estudo

A organização possui uma equipe de desenvolvimento alocada no sistema Alfa, além de equipes de outras áreas da engenharia de *software*. A equipe de desenvolvimento possui um total de quatro desenvolvedores, que trabalham exclusivamente no desenvolvimento desse sistema, eles estão representados na Tabela 6.

Tabela 6. Tabela de desenvolvedores do sistema Alfa.

Dev 1
Dev 2
Dev 3
Dev 4

Na fase que antecede o início da construção do *software* foi definido o escopo do projeto, bem como as iterações em que este se divide, caracterizado como

milestones do projeto. Dado que cada caso de uso do escopo teve uma quantidade de PFs e que a empresa possui uma base histórica de estimativas de outros projetos, então foi possível determinar a variável ***bl_remaining_time*** como exibido na Tabela 7.

Tabela 7. Tabela pontuação por Caso de Uso.

Caso de Uso	*bl_remaining_time*
UC001	80
UC002	40
UC003	45
UC004	75
UC005	30
UC006	90
UC007	15
UC008	20
UC009	85
UC010	20
UC011	40
UC012	15
UC013	15
UC014	30
UC015	12
UC016	28
UC017	20
UC018	60

Cada *milestone* possui uma quantidade determinada de casos de uso a serem desenvolvidos e cada um dos desenvolvedores da equipe se responsabiliza pela codificação de um ou mais casos de uso dentro de cada *milestone*. A Tabela 8 apresenta a divisão de casos de uso por *milestone*.

Tabela 8. Tabela casos de usos por *milestone.*

Entrega (*milestone*)	Casos de Uso
Iteração 1	UC001
	UC002
	UC003
	UC004
	UC005

Entrega (*milestone*)	Casos de Uso
	UC006
	UC007
	UC008
	UC009
Iteração 2	UC010
	UC011
	UC012
	UC013
	UC014
	UC015
	UC016
	UC017
	UC018

4.3 Desenvolvimento da Iteração 1

Ao iniciar a fase de codificação do sistema Alfa, os cinco desenvolvedores são alocados em casos de usos diferentes da Iteração 1, assim cada um deles terá como tarefas a codificação dos casos de uso que estiverem alocados em menos tempo e com maiores precisões possíveis.

Nessa Iteração a divisão de funcionalidades foi realizada de maneira que os desenvolvedores sejam alocados aos casos de uso que na soma das ***bl_remaining_time*** possuam o mesmo tamanho para os desenvolvedores. A divisão de casos de usos por desenvolvedor é apresentada na Tabela 9.

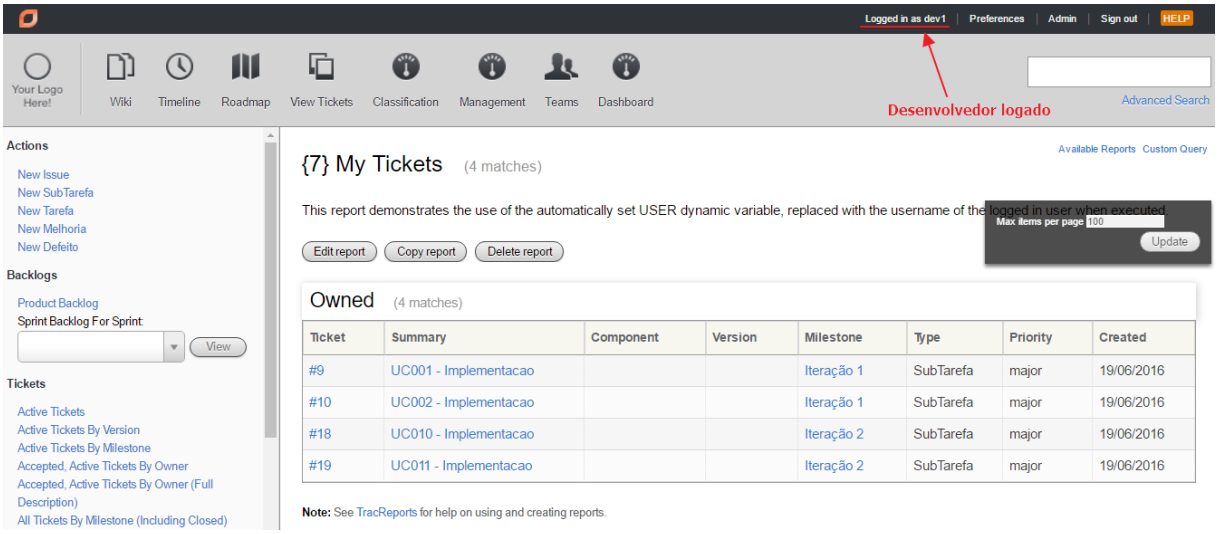
Tabela 9. Tabela casos de usos por desenvolvedor.

Desenvolvedor	Casos de Uso
Dev 1	UC001
	UC002
Dev 2	UC003
	UC004
Dev 3	UC005
	UC006
Dev 4	UC007
	UC008
	UC009

4.3.1 Andamento da Codificação

Dado início ao processo de codificação do sistema Alfa, cada desenvolvedor tem acesso a suas tarefas na tela de visualização *My Tickets* visualizado na Figura 11, onde apenas suas tarefas são exibidas. Na seleção de alguma delas ele tem acesso a informações mais detalhadas e a operações de modificação de status e de algumas das informações da tarefa.

Figura 11. Visualização de tíquetes do desenvolvedor.



Fonte: Elaborado pelo autor.

Após realizar uma análise da funcionalidade que o caso de uso propõe e qual é o ***bl_remaining_time*** do mesmo, o desenvolvedor preenche o seu ***remaining_time*** e a partir deste momento pode dar início a codificação do caso de uso. Para a situação onde o ***remaining_time*** tem uma diferença proporcionalmente grande do ***bl_remaining_time*** deve haver um acordo com o gerente do projeto para que seja no cronograma do projeto seja considerado o ***remaining_time***. As estimativas dos desenvolvedores são exibidas na Tabela 10.

Tabela 10. Tabela de estimativa dos desenvolvedores da Iteração 1.

Caso de Uso	Remaining_time
UC001	80
UC002	50
UC003	40
UC004	75
UC005	28

Caso de Uso	Remaining_time
UC006	80
UC007	30
UC008	30
UC009	84

Após iniciado o processo de codificação, todos os desenvolvedores, ao término do dia de trabalho, devem preencher a quantidade de horas trabalhadas nos casos de uso de suas reponsabilidades. Assim a porcentagem de finalização da tarefa aumenta a cada vez que suas horas são apontadas, uma vez que seu ***total_hours*** se aproxima do ***bl_remaining_time***.

No presente caso de estudo é considerado que os desenvolvedores possuem o período de jornada de trabalho semanal de 40 horas. Baseado nisso, pode-se analisar de uma forma mais abrangente na Tabela 11 o registro de horas de cada um dos desenvolvedores ao termino da Iteração 1, onde a coluna Horas registradas (por casos de uso) exibe quantidades de horas gastas na semana pelos respectivos casos de uso da coluna anterior. Além disso, durante a semana em que o desenvolvedor tenha terminado a codificação de seus casos de uso, o registro de horas em tarefas de outros projetos também será considerado.

Tabela 11. Tabela de horas registradas da Iteração 1.

Desenvolvedor	Semana	Casos de Uso	Horas registradas (por caso de uso)	Dias registrados corretamente
Dev 1	Semana 1	UC001	40	3
	Semana 2	UC001; UC002	35; 5	2
	Semana 3	UC002	30	5
	Semana 4			2 (outro projeto)
Dev 2	Semana 1	UC003	40	5
	Semana 2	UC003; UC004	12; 28	4
	Semana 3	UC004	40	5
	Semana 4	UC004	30	4
Dev 3	Semana 1	UC005; UC006	30; 10	3
	Semana 2	UC006	40	4
	Semana 3	UC006	40	4

Desenvolvedor	Semana	Casos de Uso	Horas registradas (por caso de uso)	Dias registrados corretamente
	Semana 4	UC006	8	1 + 4 (outro projeto)
Dev 4	Semana 1	UC007; UC008	30; 10	2
	Semana 2	UC008; UC009	20; 20	2
	Semana 3	UC009	40	0
	Semana 4	UC009	30	4 + 1 (outro projeto)

Terminado o desenvolvimento dos casos de usos alocados na iteração 1, as funcionalidades codificadas pelos desenvolvedores são submetidas a rodadas de testes onde são encontrados alguns defeitos de codificação e submetidos para a correção pelos desenvolvedores responsáveis pelos casos de uso dos quais se originaram os erros. Nessa situação, todos os casos de uso tiveram algum erro e com isso foi necessário gastar esforço em horas para a correção deles. Logo, ao fim da correção dos erros referente aos casos de uso da iteração 1, os desenvolvedores acumularam um total de horas utilizadas em suas correções como é exibido na Tabela 12.

Tabela 12. Tabela de esforço gasto na correção de defeitos da Iteração 1.

Caso de Uso	Esforço gasto em correção de erros (em horas)
UC001	8
UC002	12
UC003	5
UC004	20
UC005	12
UC006	2
UC007	2
UC008	2
UC009	7

4.3.2 Pontuação e Classificação

Ao finalizar a iteração 1 é possível calcular a pontuação de cada um dos desenvolvedores de acordo com a codificação dos casos de uso e correção de

defeitos provenientes deles. Com a pontuação de todos os desenvolvedores através do módulo implementado, o Agilo exibe três classificações de desenvolvedores mais produtivos. Neste momento o *ranking* de mais produtivo da *milestone* exibe os desenvolvedores ordenados decrescentemente pelos pontos ganhos na Iteração 1.

O *ranking* dos mais produtivos do projeto exibirá a mesma classificação do anteriormente referido, pois se trata da primeira *milestone* do projeto onde não há pontuação anterior a esta entrega para o projeto Alfa. O terceiro *ranking* deve apresentar a classificação de desenvolvedores mais produtivos da organização inteira, ou seja, considerando a pontuação de todos os projetos desenvolvidos pela organização, o qual não é tratado no presente estudo de caso. Antes da correção de defeitos, ao terminar a codificação dos casos de uso, a pontuação já pode ser visualizada como mostra a Figura 12. Após a correção de defeitos a classificação é alterada e a nova pontuação ao fim da iteração é vista na Tabela 13.

Figura 12. Pontuação antes das correções de erros da Iteração 1.

Fonte: Elaborado pelo autor.

Tabela 13. Pontuação ao termino da Iteração 1.

Colocação	Desenvolvedor	Pontuação
1	Dev 1	101,05
2	Dev3	87,65
3	Dev4	87,11
4	Dev2	86,34

4.3.3 Conquista de Distintivos

Ao fim da iteração, são definidos os distintivos que cada desenvolvedor recebeu por suas conquistas.

Para o distintivo de **estimador de esforços mais preciso** foi calculado que o desenvolvedor Dev 4, que apesar de não ter sido tão produtivo e gastar mais horas para finalizar suas atividades, teve uma precisão maior ao estimar seu esforço necessário para realização delas, ficando com o distintivo nessa *milestone*.

O distintivo do **desperdiçador de esforços** foi atribuído ao Dev 3, que apesar de ter pontuado bem no desenvolvimento de seus casos de uso, gastou bastantes esforços para o conserto de defeitos nos casos de uso desenvolvidos por ele.

Para o emblema de **fidelidade no registro de horas**, os desenvolvedores foram estratificados de acordo com a frequência do registro de horas. Logo, cada um tem seu *status* ao fim de cada semana e um *status* final ao término da iteração. A Tabela 14 exibe a conquista dos distintivos dos desenvolvedores na iteração 1.

Tabela 14. Tabela de conquistas de distintivos da Iteração 1.

Desenvolvedor	Desperdiçador de esforços	Estimador de esforços mais preciso	Fidelidade no registro de horas
Dev 1			Bronze
Dev 2			Diamante
Dev 4		x	Latão
Dev 3	x		Ouro

4.4 Desenvolvimento da Iteração 2

Ao iniciar a Iteração 2 os desenvolvedores foram alocados em novos casos de uso com diferentes ***bl_remaining_time***, de maneira que o tempo gasto por eles seja o mais semelhante possível, na tentativa de paralelizar a codificação dos casos de uso. A Tabela 15 apresenta a divisão das tarefas da Iteração 2 e o esforço estimado pelos desenvolvedores para a execução das tarefas.

Tabela 15. Tabela de casos de usos com tempo estimado pelos desenvolvedores.

Desenvolvedor	Casos de Uso	*Remaining_time*
Dev 1	UC010	24
	UC011	44
Dev 2	UC012	15
	UC013	20
	UC014	30
Dev 3	UC015	15
	UC016	30
	UC017	20
Dev 4	UC018	60

4.4.1 Andamento da codificação

Em continuação com processo de codificação na iteração 2, todos os desenvolvedores continuam registrando horas gastas com suas tarefas no fim da jornada diária de trabalho nos casos de uso de suas responsabilidades, como antes realizado na iteração 1. Os registros de horas da atual iteração podem ser visualizados na Tabela 16 onde a quantidade de horas desta iteração foi menor que aquela da iteração anterior devido o fato dos casos de usos restantes se caracterizarem por possuírem um menor tamanho considerando o ***bl_remaining_time*.**

Tabela 16. Tabela de horas registradas da Iteração 2.

Desenvolvedor	Semana	Casos de Uso	Horas registradas (por caso de uso)	Dias registrados corretamente
Dev 1	Semana 1	UC010; UC011	22; 18	3
	Semana 2	UC011	22	2
Dev 2	Semana 1	UC012; UC013	15; 25	5
	Semana 2	UC014	32	5
Dev 3	Semana 1	UC015; UC016	15; 25	5
	Semana 2	UC017	22	3
Dev 4	Semana 1	UC018	40	2
	Semana 2	UC018	20	5

Ao fim da codificação dos casos de usos, seguida das rodadas de teste da iteração 2, é necessário que os desenvolvedores corrijam alguns defeitos descobertos nos seus casos de uso. Ao fim da correção dos defeitos encontrados os desenvolvedores mais uma vez acumulam um total de horas utilizadas na correção desses defeitos, como mostra a Tabela 17.

Tabela 17. Tabela de Esforços gastos na correção de defeitos da iteração 2.

Caso de Uso	Esforço gasto em correção de defeitos (em horas)
UC010	5
UC012	3
UC012	5
UC013	4
UC014	2
UC015	1
UC016	3
UC017	6
UC018	8

4.4.2 Pontuação e Classificação

Ao finalizar a codificação da iteração 2 já é possível obter a quantidade de pontos e a classificação dos desenvolvedores na perspectiva do projeto e na perspectiva das iterações de maneiras diferentes. Pode-se visualizar na Figura 13 o *ranking* de pontos da iteração 2 antes da correção de defeitos.

Figura 13. Classificação dos desenvolvedores relacionados à codificação da Iteração 2.

Fonte: Elaborado pelo autor.

Após a correção de defeitos, a classificação é alterada mais uma vez e o novo *ranking* de pontos da Iteração 2, ao fim desta, pode ser visualizado na Tabela 18.

Tabela 18. Tabela de *ranking* da iteração 2.

Colocação	Desenvolvedor	Pontuação
1	Dev 4	52,0
2	Dev 1	50,18
3	Dev 3	49,14
4	Dev 2	41,13

4.4.3 Conquista de Distintivos

Assim como na iteração anterior são definidos os distintivos que os desenvolvedores conquistaram ao fim da iteração 2.

Mais uma vez o distintivo de **estimador de esforços mais preciso** foi atribuído ao desenvolvedor Dev 4, enquanto que o distintivo do **desperdiçador de esforços** foi atribuído ao Dev 2, pois este gastou um número excessivo de horas na correção de defeitos em seus casos de uso.

Para o emblema de **fidelidade no registro de horas**, os desenvolvedores foram novamente estratificados de acordo com a frequência do registro de horas,

assim como realizado na iteração 1. A Tabela 19 exibe a conquista dos distintivos dos desenvolvedores apenas na iteração 2.

Tabela 19. Tabela de conquistas de distintivos da Iteração 2.

Desenvolvedor	Desperdiçador de esforços	Estimador de esforços mais preciso	Fidelidade no registro de horas
Dev 1			Neutro
Dev 2	x		Platinum
Dev 4		x	Prata
Dev 3			Ouro

4.5 Resultados Finais

Levando em consideração que o final do projeto alfa ocorre com o término das duas *milestones* Iteração 1 e Iteração 2, alguns elementos aplicados podem ser verificados quando se finaliza o projeto.

4.5.1 Pontuação e Classificação

Como mencionado anteriormente, a pontuação e a classificação dos desenvolvedores pela pontuação também pode ser aplicada ao projeto como um todo, logo, pode-se verificar através da Tabela 20 como ficou o *ranking* de desenvolvedores mais produtivos após a finalização do projeto alfa, onde serão ranqueados pela produtividade no tempo e na qualidade do desenvolvimento, esse ultimo é considerado pela quantidade de esforço gasto no retrabalho com correção de defeitos .

Tabela 20. Tabela *ranking* do projeto Alfa.

Colocação	Desenvolvedor	Pontuação
1	Dev 1	151,23
2	Dev 4	139,11
3	Dev 3	136,79
4	Dev 2	127,47

Pode-se observar que após o desenvolvimento das duas *milestones* o desenvolvedor Dev 1 teve maior pontuação, ficando em primeiro lugar do *ranking*. Com isso, assume-se que Dev 1 foi o desenvolvedor mais produtivo da equipe.

4.5.2 Conquista de Distintivos

No final do projeto alfa, não somente os desenvolvedores considerados mais produtivos são destacados, existem outras formas de reconhecimento daqueles que não lideraram o *ranking* ao fim de um projeto, que ocorre através da atribuição de distintivos.

Após a análise das iterações as informações obtidas foram capazes de atribuir ao desenvolvedor Dev 4 o distintivo de **estimador de esforços mais preciso**, isso pode ser observado pelo bom desempenho desta habilidade nas Iterações 1 e 2.

O **desperdiçador de esforços**, como definido anteriormente, possui o intuito de mostrar as dificuldades de um desenvolvedor e incentivar que os desenvolvedores sejam mais atentos. Este aprendizado pôde ser realizado pelo Dev 2 que teve na iteração 2 um maior gasto de esforço na correção de defeitos que os demais desenvolvedores e no geral acabou sendo aquele que gastou mais horas na correção dos defeitos provenientes da codificação das funcionalidades de sua responsabilidade.

Outro distintivo que pode ser generalizado, este para cada um dos desenvolvedores, foi o de **fidelidade no registro de horas**, onde foram identificados em cada um deles o nível de prontidão com o projeto Alfa que pode ser observados, bem como os outros distintivos, na Tabela 21.

Tabela 21. Tabela de conquistas de distintivos do projeto Alfa.

Desenvolvedor	Desperdiçador de esforços	Estimador de esforços mais preciso	Fidelidade no registro de horas
Dev 1			Bronze
Dev 2	x		Platinum
Dev 4		x	Bronze
Dev 3			Ouro

4.5.3 Recompensas e Punições

Pala característica do mecanismo de recompensas e punições entende-se que são atribuições que podem variar de projeto para projeto e de empresa para empresa. Dado que existem desenvolvedores que apresentaram bons resultados no desenvolvimento do sistema, estes podem ser recompensados de alguma forma para motivá-los a sempre buscar o melhor rendimento possível. Durante o projeto Alfa em suas iterações foi possível identificar que os desenvolvedores Dev 1 e Dev 4 tiveram o que consideramos a melhor produtividade nas iterações 1 e 2, respectivamente. Por essa realização, ambos poderiam ser recompensados ao término de cada iteração. Considerando uma visão do projeto por completo, nota-se que o Dev 1 foi o mais produtivo, portanto merece um destaque maior com a atribuição de uma recompensa mais valiosa, nesse caso, a empresa alvo poderia recompensá-lo de maneiras diferentes. Uma forma de recompensá-lo é a través de produtos ou serviços de valor, como por exemplo, viagens ou produtos para uso pessoal, até mesmo pelo ato de promoção salarial do desenvolvedor, além da atribuição do distintivo de desenvolvedor mais produtivo do projeto Alfa. Outra maneira de destacar os desenvolvedores mais produtivos é a alocação dos mais bem classificados do *ranking* para o desenvolvimento de um projeto da empresa que seja considerado classe A, ou seja, um projeto mais importante que precise de bons desenvolvedores. Enquanto que os menos produtivos seriam alocados para um projeto classe B, seria aquele que não possui tanta importância para a empresa como o projeto classe A.

As recompensas também podem ser distribuídas aos desenvolvedores que foram destaque com outras habilidades durante o desenvolvimento do sistema alfa. Como foi destacado, o desenvolvedor Dev 4 como **estimador de esforços mais preciso**, este pode receber alguma premiação assim como o mais produtivo. Isso pode incentivar os funcionários a desenvolverem habilidades diferentes, logo todos possuíram funções importantes e podem ser alocados em tarefas de acordo com seu currículo de realizações dentro da empresa.

Além de recompensas, há a possibilidade da utilização do mecanismo de punição. Para o caso do projeto Alfa pode ser aplicado ao desenvolvedor que possui o *status* de **desperdiçador de esforços,** se ele tiver excedido uma quantidade

determinada de horas a ponto de fracassar ou dar prejuízos ao projeto e consequentemente à empresa. Caso o prejuízo se dê por um desenvolvedor que se mostra improdutivo (pelo *ranking* dos desenvolvedores mais produtivos), este também pode se punido, como por exemplo, presentear os desenvolvedores mais produtivos, ou ele ser o responsável por recompensar o desenvolvedor mais produtivo criando um ciclo autossuficiente de premiação dentro da empresa.

Capítulo 5
Considerações Finais

5.1 Conclusões

Promover engajamento e motivação bem como quaisquer outras experiências emocionais é um objetivo essencial da ludificação na engenharia de *software*, assim, além de tornar o ambiente de trabalho mais descontraído, pode-se haver um ganho na produtividade pelo fato dos participantes poderem desempenhar uma função que os fazem sentir bem.

Alguns elementos já utilizados em outras aplicações e que possuem um histórico de bons resultados foram aplicados ao ambiente pouco lúdico, que é o processo de desenvolvimento de *software*, e de forma específica a atender as necessidades de uma empresa que possui um processo de desenvolvimento com configurações específicas. Portanto, as funcionalidades proposta possuem traços do funcionamento real do processo de engenharia de software de uma empresa multinacional e foram adaptadas ao contexto lúdico já utilizado em trabalhos anteriores, entretanto esse modelo precisa ser avaliado no caso real.

O departamento de recursos humanos de uma empresa pode desempenhar um papel importante durante a transformação causada pelo uso da ludificação no processo de Engenharia de *Software*. É essencial fazer uma análise dos fatores emocionais que a ludificação pode afetar no trabalho dos desenvolvedores, pois a utilização de jogos nem sempre é um gosto comum de todos de uma equipe, além de que o *feedback* gerado pelos elementos de jogos pode causar constrangimento em determinadas situações.

A aplicação de um caso de estudo pôde mostrar alguns cenários onde a utilização de ludificação no Agilo *for* Trac pode fornecer informações do ambiente e promover o engajamento pela competitividade, pela atribuição de recompensas e de punições. Portanto, pôde-se notar que o *feedback* gerado a partir do uso da

ludificação tem um grande potencial para definir novas maneiras de desenvolver o processo de codificação na engenharia de *software*.

5.2 Trabalhos Futuros

Como pôde ser vista, a aplicação da ludificação através de uma ferramenta de gerenciamento de tarefas foi desenvolvida especificamente para o processo de codificação de *software*, porem pode ser estendido para as demais áreas da engenharia de *software*. Essa extensão pode ser realizada com a utilização de mais elementos de jogos tais com a criação de novos distintivos ou aplicação de novos elementos como apostas ou sistema de votação.

Como trabalho futuro, também é proposto a aplicação do módulo que implementa a ludificação no Agilo for Trac em um estudo de Caso real em diferentes empresas onde possa ser avaliado quantitativamente a contribuição dela na engenharia de *software*.

Bibliografia

[1] LOBO, E. J. R. **Curso de Engenharia de Software**. São Paulo: Digerati Books, 2008.

[2] REZENDE, D. A. **Engenharia de Software e Sistemas de Informação**. Rio de Janeiro: Brasport, 2005.

[3] PRESSMAN, R. S. **Engenharia de Software:** Uma Abordagem Profissional. São Paulo: McGraw-Hill, 2011.

[4] HUGOS, M. **Enterprise Games:** Using Game Mechanics to Build a Better Business. Sabastopol: O'Reylli, 2012.

[5] BYRON REEVES, . L. R. **Total Engagement:** Usin Games and Virtual Worlds to Change the Way People Work and Businesses Compete. Boston: Havard Business Press, 2009.

[6] CONDÉ, L. Introdução ao Application Lifecycle Management (ALM). **Microsoft**, 2009. Disponivel em: <https://msdn.microsoft.com/pt-br/library/ee156630.aspx?f=255&MSPPError=-2147217396>. Acesso em: 29 Junho 2016.

[7] NOYES, B. Rugby, Anyone?: Scrum tackles software development from a patterns perspective to turn. **http://www.fawcette.com/resources/managingdev/methodologies/scrum/**, Junho 2002.

[8] PEREIRA, J. R.; FALCONIERI, G.; ZANINI, R. Gerenciamento de projetos com Scrum e Agilo. **Engenharia de Software Magazine**, n. 71.

[9] AGILO for trac - Documentation. **Agilo for trac**. Disponivel em: <http://www.agilofortrac.com/documentation/>. Acesso em: 23 Junho 2016.

[10] YOHANNIS, A. R.; PRABOWO, D. Y.; WAWORUNTU, A. Defining Gamification: From lexical meaning and process viewpoint towards a gameful reality. **International Conference on Information Technology Systems and Innovation**, Bangdung, 24-27 Novembro 2014. 284-289.

[11] ZICHERMANN, G.; CUNNINGHAM, C. C. **Gamification by Design:** Implementing Game Mechanics in Web and Mobile Apps. 1ª. ed. Gravenstein Highway North: O'reilly, 2011.

[12] PEDREIRA, O. et al. Gamification in software engineering - A systematic mapping. **Information and Software Technology**, A Coruña, 57, 2015. 157-168.

[13] MELO, A. A. D. et al. Version Control System Gamification: A Proposal to Encourage the Engagement of Developers to Collaborate in Software Projects. In: MEISELWITZ, G. **Social Computing and Social Media**. Joinville: Springer International Publishing, v. 8531, 2014. p. 550-558.

[14] PASSOS, E. B. et al. Turning Real-World Software Development into a Game. **2011 Brazilian Symposium on Games and Digital Entertainment**, Salvador, 2011. 260-269.

[15] REDCRITTER Tracker - Home. **RedCritter Tracker**. Disponivel em: <https://www.redcrittertracker.com/home.aspx>. Acesso em: 24 Junho 2016.

[16] VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de Função:** Medição, Estimativas e Gerenciamento de Projetos de Software. 12ª Edição Revisada. ed. São Paulo: Érica, 2012.

[17] REDCRITTER Tracker: Gamifying agile project management. **Gigaom**. Disponivel em: <https://gigaom.com/2011/07/22/redcritter-tracker-agile-project-management-with-gamification/>. Acesso em: 24 Junho 2016.

Apêndice A

Script de identificação dos desenvolvedores mais produtivos após a codificação de uma *milestone*

O *Script* desta seção foi aplicado para a obtenção do *ranking* de pontos dos desenvolvedores por suas produtividades após a codificação dos casos de uso dentro de cada *milestone.*

```
SELECT
        jogador, round(sum(pontos), 2) AS total
FROM (
        SELECT
                tusuario.horas_ticket, tusuario.horas_usuario AS jogador,
                tusuario.total_hours, tcustom.value, (tcustom.value /
                (tusuario.total_hours / tcustom.value)) AS pontos
        FROM (
                SELECT
                        horas.ticket AS horas_ticket, horas.usuario AS horas_usuario,
                        MAX(horas.soma) total_hours
                FROM (
                        SELECT
                                ticket, author AS usuario, SUM(CAST(newvalue AS float))
                                AS soma from ticket_change
                        JOIN
```

```
                ticket ON (ticket_change.ticket = ticket.id)
            WHERE
                ticket_change.field='hours' AND
                ticket.type='task' AND
                ticket.status='closed' AND
                ticket.milestone='nome da milestone' AND
                ticket.resolution='fixed' AND
                summary like '%Implementacao%'
            GROUP BY
                author, ticket) horas
        GROUP BY
            horas_ticket) tusuario
    JOIN
        ticket_custom tcustom ON (tcustom.ticket = tusuario.horas_ticket)
    WHERE
        name = 'bl_remaining_time')
GROUP BY
    jogador
ORDER BY
    total desc
    .
```

Apêndice B
Script de identificação do desenvolvedor estimador de esforços mais preciso

O *Script* desta seção foi aplicado para a obtenção do distintivo de **estimador de esforços mais preciso**.

```
SELECT
        horas_usuario AS Estimador, MIN(erro) AS variancia
FROM (
        SELECT
                horas_usuario, AVG(ABS((total_hours / remaining_time) - 1)) AS erro
        FROM (
                SELECT
                        horas.ticket AS horas_ticket, horas.usuario AS horas_usuario,
                        MAX(horas.soma) total_hours
                 FROM (
                        SELECT
                                ticket, author AS usuario, SUM(newvalue) AS soma
                        FROM
                                ticket_change, ticket
                        WHERE
                                ticket_change.ticket = ticket.id AND
                                ticket_change.field='hours' AND
                                ticket.type='task' AND
                                ticket.status='closed' AND
```

```
                ticket.resolution='fixed' AND
                summary like '%Implementacao%'
        GROUP BY
                author, ticket) horas
GROUP BY
        horas_ticket) ownerr
JOIN (
        SELECT
                ticket, MIN(CAST(oldvalue AS float)) AS remaining_time
        FROM
                ticket_change, ticket
        WHERE
                ticket_change.ticket = ticket.id AND
                ticket_change.field='remaining_time' AND
                ticket.type='task' AND
                ticket.status='closed' AND
                ticket.resolution='fixed' AND
                summary like '%Implementacao%'
        GROUP BY
                ticket) remaining
ON
        ownerr.horas_ticket = remaining.ticket
GROUP BY
        horas_usuario)
```