UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA
MBA EM GESTÃO DE NEGÓCIOS E TECNOLOGIA DA INFORMAÇÃO

RODRIGO KLEIN

PROPOSTA DE GAMIFICAÇÃO PARA REDUÇÃO DE RETRABALHO: ESTUDO NA EMPRESA SYSTEMHAUS

São Leopoldo
2015

Rodrigo Klein

PROPOSTA DE GAMIFICAÇÃO PARA REDUÇÃO DE RETRABALHO: ESTUDO NA EMPRESA SYSTEMHAUS

Trabalho de Conclusão de Curso de Especialização apresentado como requisito parcial para obtenção do título de Especialista em MBA em Gestão de Negócios e Tecnologia da Informação da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof.(a) MS Rosemary Francisco

São Leopoldo

2015

Dedico este trabalho aos meus filhos que esperaram pacientemente que concluísse o mesmo antes de mudarem minha vida para sempre.

AGRADECIMENTOS

Para que a realização deste trabalho fosse possível contei com a compreensão e apoio de diversas pessoas, as quais agradeço profundamente.

Agradeço a minha esposa pelo apoio em todos os momentos da minha vida.

Aos meus pais e meu irmão por sempre me lembrarem da importância do estudo e pelo incentivo para concluir este trabalho.

A minha orientadora, Rosemary Francisco, pela dedicação, mesmo quando mandava quatro versões do trabalho na mesma semana.

Aos professores do MBA que contribuíram muito para meu crescimento pessoal e profissional.

Aos colegas de curso pelas mensagens de incentivo ou descontração durante a realização desta pesquisa.

E por fim a empresa e colegas de trabalho que se dispuseram a auxiliar no desenvolvimento da pesquisa, com suas opiniões e tempo dispendido.

RESUMO

Empresas de *software* encontram no retrabalho um de seus maiores custos, seja em um novo projeto ou na manutenção de um produto existente. As causas destes retrabalhos podem ter diversas origens, porém muitas delas são evitáveis e apresentam uma oportunidade para as empresas reduzirem seus custos operacionais e concluírem seus projetos dentro do prazo e ao mesmo tempo com maior qualidade. Para tal, as empresas podem utilizar técnicas e processos ou mesmo motivar os indivíduos para que sejam mais focados e cuidadosos ao atender as solicitações. Gamificação é uma abordagem onde se faz uso de elementos de jogos para alterar o comportamento das pessoas. Esta técnica tem sido utilizada em diversos contextos, como treinamentos, fidelização de usuários ou mesmo nos processos de trabalho. O que esse trabalho apresenta é uma proposta para motivar times responsáveis pelo desenvolvimento e manutenção de um produto de *software*, e que utilizam o *framework Scrum*, a fazerem uso de certas técnicas (como testes unitários e refatoração) e buscarem a excelência em seu dia a dia através de gamificação. Essa pesquisa é um estudo de caso único, no qual aplicaram-se os procedimentos de grupo focal e análise de documentos para compreensão do cenário. Entre os principais resultados destacam-se a importância da motivação na equipe de desenvolvimento de software para evitar retrabalhos, e técnicas de desenvolvimento de software que acabam resultando em um menor índice de retrabalhos. Outro resultado importante do grupo focal realizado é que a gamificação pode ser utilizada no processo para buscar um incremento motivacional e, assim, reduzir o número de retrabalhos.

Palavras-chave: *Software.* Retrabalho. Gamificação. *Scrum*.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTA DE SIGLAS

PDCA	*Plan, Do, Check and Act*
PSP	*Personal Software Process*
SDT	*Self-determination theory*

SUMÁRIO

1 INTRODUÇÃO

Apesar de diversos estudos como os de Toor e Ogunlana (2010) e Atkinson (1999) já apontarem novos fatores, alinhados com a estratégia do negócio, para se avaliar o sucesso de um projeto; tempo, qualidade e custo são ainda fundamentais como indicadores de sucesso. O tempo é essencial para uma equipe de desenvolvimento de software que atenda diversos projetos concorrentes e ainda realiza a manutenção de um produto.

O retrabalho no desenvolvimento de software é toda a alteração que deve ser executada em uma versão já existente do software (FAIRLEY; WILLSHIRE, 2005). Estas alterações podem ser planejadas ou não, e no caso de um projeto, os retrabalhos não planejados podem afetar o tempo e o custo do projeto como um todo, uma vez que são inesperados.

Desta forma a redução de retrabalho é fundamental no sentido de evitar atrasos e custos extras durante a execução de um projeto. De acordo com Chua (2010), uma boa compreensão das alterações de requisitos em relação ao custo do retrabalho é extremamente importante. Devido a sua característica inconstante, a mesma pode gerar um efeito cascata obrigando novas alterações, sendo que seu impacto pode possivelmente dobrar o custo total previsto inicialmente.

Boehm & Papaccio (1988) apontam a necessidade de se identificar as alterações e correções em fases iniciais do processo, pois o custo gerado pelas alterações aumentam conforme o projeto avança.

O retrabalho tem origem em pedidos de alteração do *software* ou solicitações de correções, essas solicitações são feitas por diversos tipos diferentes de pessoas. Stark *et al.* (1999, p. 294) afirmam que

> No ambiente de manutenção, os requisitos são recolhidos através de solicitações de mudanças que identificam novos recursos e / ou correções de defeitos. As solicitações de mudanças são geradas a partir de uma variedade de pessoas, incluindo tomadores de decisão, operadores de sistemas, desenvolvedores e equipes externas. Essas pessoas têm diferentes origens e níveis de compreensão em relação a computadores e operações do sistema. Esta diversidade, muitas vezes leva a uma má interpretação da intenção e do âmbito da solicitação de mudança pela equipe de manutenção, resultando em um requisito incompleto ou incorreto. Esta má interpretação impacta no dimensionamento do esforço associado com a implementação do requisito. Além disso, durante o processo de implantação, os requisitos mudam frequentemente.

Chua (2010) identificou diversos estudos que apontam que alguns dos principais fatores de alteração em um *software* estão relacionados ao ambiente e a organização. Trabalhos que detalham como a falta de motivação, a falta de habilidades, o conhecimento inadequado e a falta de experiência do desenvolvedor afetam as atividades durante a manutenção de um *software*.

Considerando o efeito que o retrabalho tem em um ambiente de desenvolvimento de *software* que executa diversos projetos em paralelo com o objetivo de adaptar um produto existente, e tendo em vista que alguns dos fatores que influenciam a demanda destes retrabalhos estão relacionados com a motivação da equipe de desenvolvimento e o conhecimento da mesma (CHUA, 2010 apud MOLOKKEN; JORGENSEN, 2003) este trabalho tem como objetivo apresentar uma proposta utilizando gamificação para buscar a redução da quantidade de retrabalho gerado em uma equipe pequena de desenvolvimento responsável por projetos de evolução e manutencão de um produto. Esta técnica de gamificação é definida por Deterding *et al.* (2011) como “o uso de elementos de jogos em contextos não relacionados a jogos”.

1.1 SITUAÇÃO PROBLEMA E PERGUNTA DE PESQUISA

Boehm & Basili (2001) afirmam que projetos de software atuais consomem entre quarenta e cinquenta por cento de seu esforço em retrabalho que poderia ter sido corrigido em um estágio inicial ou mesmo evitado por completo.

O retrabalho evitável, quando se apresenta em uma quantidade excessiva, impacta em atrasos de cronograma e desmoraliza a equipe, reduzindo sua motivação. Além disso, os custos aumentam e a produtividade reduz, a confiança do cliente/usuário também é afetada negativamente pois grande parte deste retrabalho surge de defeitos identificados pelo mesmo (FAIRLEY; WILLSHIRE, 2005). Um retrabalho com o objetivo de corrigir um problema encontrado após a entrega do *software* pode resultar em um custo até cem vezes maior do que se o problema tivesse sido identificado em uma fase inicial do projeto (BOEHM; BASILI, 2001).

Westland (2004, p. 237) estudou o comportamento do custo de defeitos de *software* e concluiu que “[...] não apenas a detecção e correção de erros contribui em proporção substancial ao custo total do *software*, mas que o gerenciamento da detecção e correção pode ser complexo. Neste mesmo trabalho, Westland(2004)

comprova a natureza de crescimento exponencial do custo da correção de um defeito em relação a fase em que o mesmo é identificado.

A empresa que será objeto de estudo neste trabalho é a SystemHaus S.A. Fundada em Novo Hamburgo em 1988, a SystemHaus foi idealizada para proporcionar soluções informatizadas sob demanda a uma carteira de clientes locais. Com o passar dos anos o foco passou a ser desenvolvimento de soluções exclusivas para a indústria do couro e expandiu-se a área de atuação, hoje atendendo clientes no mundo todo. Apesar dessa abrangência mundial a empresa é de pequeno porte e atende seus clientes com uma equipe de desenvolvimento também pequena, atualmente com oito desenvolvedores.

Esta empresa se encontra em uma situação que representa exatamente o cenário foco deste trabalho que é avaliar a utilização de gamificação no processo de desenvolvimento de pequenas empresas com o objetivo de redução de retrabalho. Desta forma, o estudo será limitado aos retrabalhos evitáveis causados por defeitos, pois estes retrabalhos podem ser evitados durante o processo de desenvolvimento.

Uma vez exposto esta situação, a questão-problema que direciona este trabalho é: **Como Gamificação pode ser utilizado para reduzir o retrabalho em equipes de desenvolvimento pequenas que atendam evoluções e manutenção de software?**

1.2 OBJETIVOS

Os objetivos geral e específicos para atender a questão de pesquisa proposta por este trabalho estão listados nos dois subcapítulos seguintes.

1.2.1 Objetivo Geral

Analisar como gamificação pode ser utilizada para a redução de retrabalho em equipes pequenas de desenvolvimento e manutenção de software.

1.2.2 Objetivos Específicos

Os objetivos específicos para atender o objetivo geral são:

a) Identificar as possíveis causas da ocorrência de retrabalho durante o desenvolvimento de software;
b) analisar abordagens de gamificação utilizadas no contexto de desenvolvimento e manutenção de software;
c) propor uma abordagem utilizando gamificação para reduzir o retrabalho no contexto de desenvolvimento e manutenção de software.

1.3 JUSTIFICATIVA

O retrabalho representa um custo bastante grande para o desenvolvimento de *software*, podendo representar em certos casos 50% do esforço total de um projeto ou do tempo gasto em manutenção de um *software* (BOEHM; BASILI, 2001). Por isso qualquer esforço para reduzir essa quantidade é bem vinda para qualquer empresa de desenvolvimento.

Outro problema causado pelo retrabalho são os desvios em estimativas de tempo, que para algumas empresas é fundamental para definir valores de contratos ou produtos (AFSHARIAN; GIACOMOBONO; INVERARDI, 2008).

Ainda, segundo Afsharian, Giacomobono e Inverardi (2008), a principal causa do retrabalho está relacionado com os defeitos encontrados durante a fase de desenvolvimento. Este retrabalho pode, por exemplo, ter origem em um processo que não está conseguindo prevenir o mesmo, pode estar relacionado a falta de habilidades por partes dos desenvolvedores ou mesmo a falta de motivação destes (FAIRLEY; WILLSHIRE, 2005).

Nesse cenário o conceito de gamificação parece se encaixar perfeitamente, uma vez que utiliza elementos de jogos em diferentes contextos (DETERDING; DIXON, 2011) com o intuito de impactar positivamente no engajamento de um individuo no contexto gamificado (DUVERNET; POPP, 2014).

Segundo Rajamarthandan (2014), a gamificação pode ser uma das ferramentas utilizadas para se buscar qualidade de software, entregas na data estimada, inovações, etc., principalmente afetando a motivação do funcionário.

O objetivo final desta pesquisa é com base no que foi identificado sobre o retrabalho em uma equipe de desenvolvimento de software de pequeno tamanho, associado com o que foi verificado na literatura sobre o assunto, propor uma abordagem de uso de gamificação com o intuito de reduzir estes índices de

retrabalho e ainda verificar a opinião da equipe da unidade de estudo sobre a efetividade da abordagem proposta.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo tem como objetivo contextualizar assuntos como desenvolvimento de software, retrabalho e gamificação.

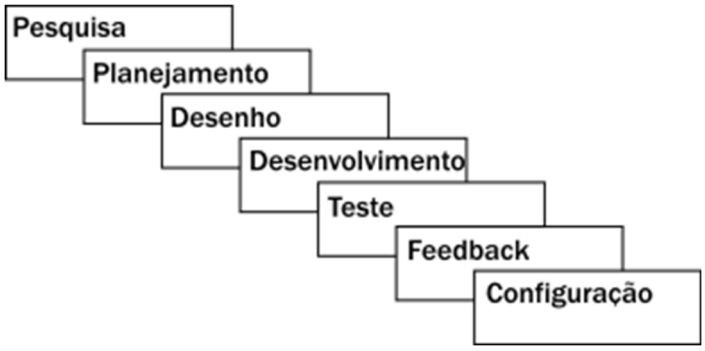
2.1 DESENVOLVIMENTO DE SOFTWARE

Para o desenvolvimento desta pesquisa é fundamental entender como ocorre o processo de desenvolvimento de um software.

O processo de desenvolvimento de um software é dividido em diversos estágios que representam o ciclo de vida do processo (DAWSON; DAWSON, 2014). Dawson e Dawson (2014) definem como principais etapas do desenvolvimento de software a análise de requisitos, o desenho da solução, a construção do software, a etapa de testes e por fim a implementação do mesmo. Estas etapas são executadas em qualquer desenvolvimento de software, com intensidades diferentes dependendo do processo adotado pela equipe.

Despa (2014) adiciona mais dois estágios em um ciclo genérico de desenvolvimento de software que são o planejamento, que ocorre após o levantamento de requisitos, e a manutenção que ocorre no final do processo. A nomenclatura utilizada por Despa (2014) difere um pouco da nomenclatura utilizada por Dawson e Dawson (2014), e seu processo genérico é representado na Figura 1.

Figura 1 – Processo genérico de desenvolvimento de software



Fonte: Adaptado de Despa (2014)

Dawson e Dawson (2014) definem cinco modelos fundamentais de processo que representam todos os modelos existentes. São eles:

- Constrói e Corrige: Dawson e Dawson (2014) definem este modelo como não estruturado, e todos os estágios são sobrepostos sem definição clara de cada estágio.

- Cascata: Processos que seguem estes modelos, em geral, separam cada etapa do processo claramente. A etapa seguinte só é executada quando a anterior foi completamente concluída (DAWSON; DAWSON, 2014).

- Incremental: No processo incremental o software é dividido em diversas entregas incrementais. Os requisitos são levantados no inicio do processo e o desenho da solução, a construção e os testes são executados em ciclos com entregas parciais. Essa repetição ocorre até a construção completa do software (DAWSON; DAWSON, 2014).

- Evolutivo: De acordo com Dawson e Dawson (2014) o processo evolutivo é muito semelhante ao processo incremental, a diferença fundamental entre os dois é que no processo evolutivo os requisitos também são reavaliados a cada ciclo.

- Protótipo: Este modelo se assemelha ao processo em cascata, uma vez que as etapas são claramente separadas. O diferencial deste processo é que antes de se iniciar o desenvolvimento propriamente, um protótipo é criado para se validar alguma inferência feita durante o desenho da solução ou para o esclarecimento de dúvidas sobre os requisitos. Após o protótipo ter atendido seu objetivo, o mesmo é descartado (DAWSON; DAWSON, 2014).

A unidade de análise observada por esta pesquisa utiliza o *framework* de desenvolvimento de software chamado Scrum, que de acordo com Dawson e Dawson (2014) é um cruzamento entre o modelo incremental e o modelo evolutivo. No tópico que se segue é apresentado mais detalhes sobre este *framework*.

2.1.1 SCRUM

Dawson e Dawson (2014) consideram o Scrum como uma abordagem que está posicionada entre o modelo incremental e o modelo evolutivo. Isso porque, apesar de não possuir uma etapa de reavaliação dos requisitos, o processo aproxima o cliente da equipe de desenvolvimento, permitindo a alteração e o refinamento dos requisitos entre cada ciclo incremental.

O Scrum não é um processo ou uma técnica, mas sim um *framework* de processo onde diversos processos e técnicas podem ser aplicadas. Este *framework* tem como base a teoria empírica de que o conhecimento é fruto da experiência e da tomada de decisão baseada em o que se conhece (SUTHERLAND; SCHWABER, 2013).

Sutherland e Schwaber (2013) afirmam que o Scrum tem como pilares a transparência, onde tudo que for significativo do processo deve estar visível; a inspeção, onde o time deve avaliar os artefatos gerados frequentemente; e a adaptação, que ocorre quando durante a inspeção se identifica variações inaceitáveis e o processo/produto deve ser adaptado para corrigir o desvio.

O Scrum define três papéis em um time, o Scrum Master, que tem a responsabilidade de garantir que o processo seja seguido, que as cerimônias (eventos) propostas pelo *framework* sejam realizadas e também deve remover os impedimentos da equipe, permitindo que a mesma consiga realizar o seu trabalho. O *Product Owner* que deve garantir que o trabalho executado está gerando valor para o produto/negócio, bem como definir as prioridades das histórias que deveriam estar no próximo *sprint*. E por fim, o time de desenvolvedores, que é auto organizado e multifuncional e tem como objetivo a entrega do produto pronto (SUTHERLAND; SCHWABER, 2013).

Além dos papéis, o Scrum define alguns eventos que ocorrem durante o processo, estes eventos possuem sempre uma duração máxima para ocorrer. O *Sprint* é o principal evento e tem duração máxima de um mês (podendo variar de time para time), é neste evento que o incremento do software é realizado e dentro dele ocorrem alguns dos outros eventos (SUTHERLAND; SCHWABER, 2013).

O evento chamado *Sprint Planning* ocorre antes de o *Sprint* iniciar, e neste evento é definido, de forma colaborativa entre os elementos do time, um plano do que será entregue no fim do *Sprint* (SUTHERLAND; SCHWABER, 2013).

A reunião diária ocorre durante o *Sprint* e é um evento bem curto (em média 15 minutos) que tem como objetivo inspecionar o trabalho realizado nas últimas 24 horas e planejar o próximo dia de trabalho (SUTHERLAND; SCHWABER, 2013).

A revisão do *Sprint* ocorre no final de um *Sprint* e é onde é feita a inspeção do trabalho realizado no *Sprint* e a revisão e adaptação do *Backlog* de produto, já o preparando para o próximo *Sprint* (SUTHERLAND; SCHWABER, 2013).

O último evento previsto no Scrum é a retrospectiva do *Sprint*, onde o time inspeciona o próprio processo, identificando problemas e propondo soluções. Este evento é específico para inspeção e adaptação, porém quaisquer melhorias podem ser adotadas a qualquer momento, sem a necessidade de aguardar este evento (SUTHERLAND; SCHWABER, 2013).

Sutherland e Schwaber (2013) definem três artefatos no *framework*, o *backlog* de produto, que define os itens que farão parte do produto ordenados por valor que os mesmos representam, o *backlog* do *Sprint*, que contém uma fatia do *backlog* de produto identificando os itens que serão concluídos no *Sprint* atual, e o incremento, que é a soma dos itens do *backlog* de produto que estão realmente sendo entregues no Sprint *atual*.

Mesmo com o uso de um framework como o *Scrum*, durante o processo de desenvolvimento de software ocorrem erros ou mesmo alterações de escopo, o que acaba gerando retrabalho. O capitulo seguinte apresenta a definição de retrabalho e seu efeito no custo de um software.

2.2 RETRABALHO E CUSTO DE SOFTWARE

Conforme Fairley e Willshire (2005) retrabalho pode ser definido como qualquer alteração que ocorra em uma das quatro dimensões que caracterizam um software. São elas:

- funcionalidade: são as características computacionais que fornecem a funcionalidade do software;

- estrutura: é a estrutura em si, envolvendo o programa e os dados, bem como as interconexões dinâmicas que ocorrem entre eles;

- comportamento: representa como o software altera seu estado conforme a ocorrência de eventos internos ou externos;

- atributos de qualidade: inclui performance, confiança, segurança, facilidade de modificação e reusabilidade.

De uma forma mais simples, Cass, Osterweil e Wise (2009, p. 305) definem retrabalho como “[...] a atividade de reconsiderar e modificar uma decisão anterior [...]”.

Segundo Afsharian, Giacomobono e Inverardi (2008) o retrabalho é uma das principais causas dos desvios entre o que foi realizado e o que foi estimado durante o desenvolvimento de software.

O retrabalho se divide em evitável e inevitável sendo que estudos indicam que entre 40 e 50 porcento do trabalho gasto em um projeto de software está relacionado com retrabalho evitável. O retrabalho inevitável está se tornando mais

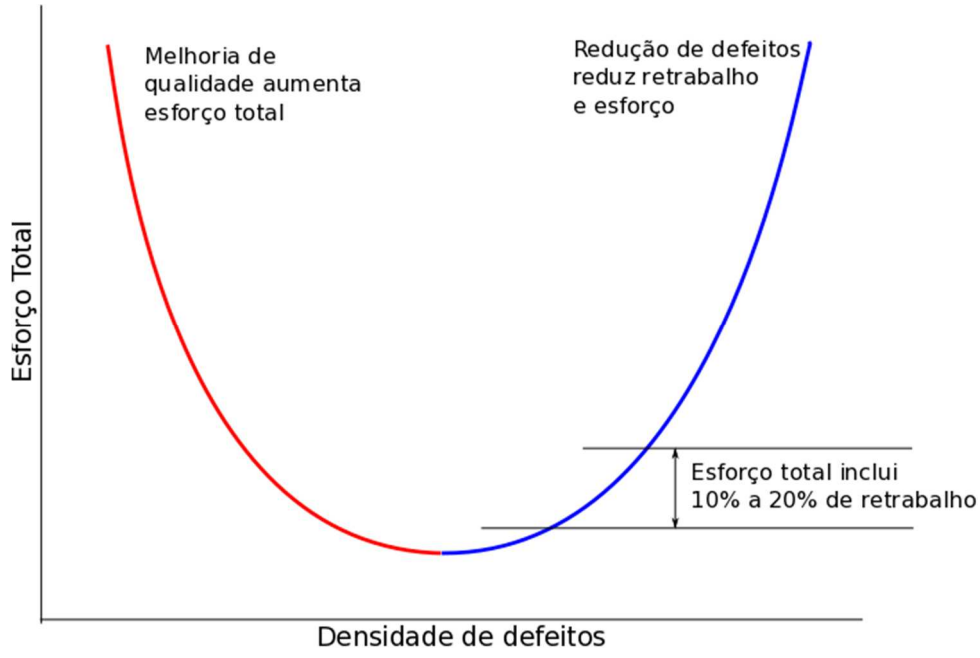
comum com a evolução de processos iterativos, onde protótipos são criados e alterados a cada iteração (BOEHM; BASILI, 2001).

Retrabalhos causados pela evolução do software são considerados inevitáveis pois sua ocorrência está relacionada a algo que até então não havia sido definido, já os retrabalhos evitáveis estão relacionados a duas categorias, retrabalhos de retrospectiva onde o desenvolvedor optou por não desenvolver determinada característica mesmo sabendo que a mesma seria necessária mais adiante e retrabalhos corretivos que estão ligados a correções de defeitos não esperados (FAIRLEY; WILLSHIRE, 2005).

Fairley e Willshire (2005) apontam a necessidade de se coletar dados de retrabalho e ainda categorizá-los em um dos três tipos (evolucionário, de retrospectiva ou corretivo). Porém, indicam que nem sempre esta categorização é simples, pois muitas vezes o ponto de vista de um desenvolvedor pode ser diferente do ponto de vista do requisitante.

Ainda, segundo Fairley e Willshire (2005) quando o retrabalho evitável ocorre excessivamente acaba impactando negativamente no cronograma (gerando atrasos) e afetando a motivação dos funcionários. O cliente também acaba sendo afetado, pois muitas vezes o retrabalho é causado por defeitos identificados por ele e sua confiança na solução acaba reduzindo. A Figura 2 demonstra a variação de esforço necessário em relação a quantidade de defeitos encontrados no desenvolvimento.

Figura 2 – Esforço poupado reduzindo defeitos



Fonte: Adaptador de Fairley e Willshire (2005)

Identificar e corrigir as causas raiz dos retrabalhos evitáveis oferecem uma ótima oportunidade para aumentar a qualidade, produtividade, moral dos desenvolvedores e a satisfação do cliente (FAIRLEY; WILLSHIRE, 2005).

Boehm e Basili (2001) identificaram, inclusive, que os ganhos relacionados a melhorias de processo, arquitetura de software e gerenciamento de riscos em sua grande maioria são gerados exatamente por uma redução do retrabalho evitável.

Esse trabalho tem como foco principal estudar os retrabalhos ligados aos defeitos que passaram da etapa de desenvolvimento e portanto, são considerados retrabalhos evitáveis da categoria corretivo.

Westland (2004) ressalta em seu estudo que a detecção e a correção de defeitos tem uma grande representatividade no custo total de um software. Além do custo da própria correção, esta mesma correção, dependendo da complexidade, pode acabar inserindo novos defeitos ao software aumentando ainda mais o custo do defeito original.

Jones (2004) ainda afirma que o controle de qualidade é o fator mais importante para se alcançar o sucesso de um projeto pois a correção de erros é a tarefa que mais consome tempo e a mais cara em sistemas grandes.

Boehm e Papaccio (1988) apontam a importância da detecção nas fases iniciais do desenvolvimento do produto, pois o custo da correção aumenta em fases mais avançadas do processo. Segundo Boehm e Basili (2001), esse custo pode aumentar em fatores de até 100:1 em sistemas de alta complexidade e críticos. Entretanto, segundo os autores, este fator pode ser reduzido utilizando-se boas práticas na arquitetura do sistema. O crescimento exponencial do custo da correção de um defeito, em relação a fase em que o mesmo é identificado, também foi identificado por Westland (2004).

Técnicas modernas de programação, modularização, bem como desenvolvimento incremental, prototipação rápida e processos mais enxutos de desenvolvimento são abordagens que podem ser utilizadas com o intuito de reduzir o retrabalho (BOEHM; PAPACCIO, 1988).

Outro problema que pode gerar retrabalho no desenvolvimento é código de baixa qualidade, que conforme Martin (2008) ainda reduz a produtividade do desenvolvedor pois as alterações acabam causando problemas em trechos não esperados. Uma abordagem para resolver este problema envolve o uso de refatoração. Refatoração significa reestruturar um determinado código-fonte com o

intuito de torná-lo mais fácil de entender e de alterar sem modificar (ou muito pouco) seu comportamento observável (FOWLER, 2014). Além disso, Fowler (2014) afirma que a refatoração também ajuda a localizar falhas em códigos existentes, pois o desenvolvedor que executa a refatoração precisa compreender profundamente o que o código está tentando realizar.

A adição de práticas para disciplina pessoal também podem reduzir a taxa de introdução de novos defeitos ao software (BOEHM; BASILI, 2001). O PSP (*Personal Software Process*) é um exemplo de prática que poderia ser utilizada com este objetivo. O PSP foi desenvolvido para ser utilizado individualmente por engenheiros de software e é baseado em alguns princípios de planejamento e qualidade que tem como um dos objetivos corrigir defeitos o mais cedo possível ou até mesmo prevenir a ocorrência do defeito em si (HUMPHREY, 2000).

Apesar de ferramentas terem sido criadas e o ciclo de desenvolvimento ter sido aperfeiçoado para reduzir a ocorrência de defeitos e por consequência de retrabalho, Conroy e Kruchten (2012) apontam a necessidade de focar também no elemento humano do processo, e que seu estado mental é tão importante quanto as técnicas e ferramentas utilizadas para a busca da redução de retrabalho.

No próximo capitulo será apresentado o conceito de gamificação, o qual é uma abordagem conhecida por disponibilizar elementos de jogos em diferentes contextos para alavancar o engajamento e a motivação das pessoas em atividades como resolução de problemas.

2.3 GAMIFICAÇÃO

Detering *et al.* (2011) define gamificação como técnicas onde se utilizam elementos que são comuns em jogos em ambientes não relacionados a jogos. Desta forma, diferencia-se gamificação de *serious games* que são jogos onde o principal objetivo é, por exemplo, treinar ou educar através do mesmo.

Já Huotari e Hamari (2012) definem gamificação, por meio de uma perspectiva de marketing de serviço, como um processo de aperfeiçoamento de um serviço, adicionando qualidades que contribuam para que o usuário vivencie experiências que teriam com um jogo (*gameful experiences*), com o objetivo de apoiar a criação de valor para o usuário. Hamari (2012) explica que não existe um sinônimo para *gameful experiences* e que não existe um consenso de quais são as

experiências que se tem somente em jogos. Ainda segundo Hamari (2012), alguns psicólogos sugerem algumas características como sendo parte de uma *gameful experience*, exemplos são: maestria, autonomia, suspense, etc. E alcançar este tipo de experiência é o objetivo de qualquer aplicação de gamificação de acordo com a definição de Huotari e Hamari (2012).

E por fim, Robson *et al*. (2015) afirmam que gamificação é a aplicação de lições aprendidas no mundo dos jogos para alterar comportamentos em um contexto fora dos jogos. Afirma ainda que este tipo de aplicação pode focar tanto em processos de negócio quanto em resultados, e, pode contar com a participação de elementos de fora da empresa (clientes e fornecedores) ou de dentro dela.

O que se percebe é que apesar das definições possuírem diferenças, principalmente em relação a serem mais ou menos restritivas ao que cada autor considera uma aplicação de gamificação, todas tem como base o uso de elementos que são comuns em jogos com a finalidade de modificar ou reforçar algum comportamento.

Com base nestas definições empresas tem adotado técnicas de gamificação com frequência para influenciar pessoas a adotarem novos comportamentos, agilizar a aprendizagem de novas técnicas e tecnologias e facilitar a execução de tarefas tediosas e repetitivas (VIANNA et al., 2013).

Alguns dos benefícios apresentado por Rajamarthandan (2014) no uso de gamificação em uma organização são:

- adoção de novas habilidades e processos, utilizando gamificação para adaptar o time a essa nova habilidade ou processo. “Gamificação pode ajudar a criar novas ideias de produto/processo e soluções e descobrir informações de maneira rápida e fácil” (RAJAMARTHANDAN, 2014, p. 5);

- motivação do empregado: “Executar competições gamificadas melhora a motivação dos funcionários e será visto como uma ferramenta de reconhecimento de talentos” (RAJAMARTHANDAN, 2014, p. 5);

- colaboração: “Gamificação ajuda a manter os membros do time conectados com outros membros e a serem mais sociáveis” (RAJAMARTHANDAN, 2014, p. 5);

- *feedback* em tempo real: Reduzir o tempo de *feedback* além de motivar o time, pode evitar que o retrabalho seja criado em etapas mais avançadas do projeto, o que causa um efeito negativo maior.

No capitulo seguinte serão apresentados os elementos existentes em jogos que podem ser utilizados com o objetivo de “gamificar” um determinado processo.

2.3.1 Elementos de jogos

A definição do que é considerado um elemento de jogo não é clara, uma vez que poucos elementos são únicos e exclusivos de jogos, desta forma define-se que são aqueles que estão presentes na grande maioria dos jogos e possuem um importante papel na jogabilidade em si (DETERDING et al., 2011).

Em sua revisão literária, Hamari, Koivisto e Sarsa (2014) elencaram os elementos que mais são citados em implementações de gamificação, são eles: pontos, quadro de líderes, medalhas, níveis, história, objetivos claros, *feedback*, recompensas, progresso e desafio.

Estes elementos abrangem a mecânica e a dinâmica de jogo. A mecânica é a aplicação de componentes como se fossem blocos que podem ser utilizados para “gamificar” determinado processo. Já a dinâmica são os efeitos subjetivos na experiência do usuário que são causados por determinado elemento mecânico e correspondem a um determinado motivo (BLOHM; LEIMEISTER, 2013). O Quadro 1 apresenta alguns dos principais elementos e sua mecânica de jogo relacionada com a dinâmica e seu respectivo motivo conforme listado por Blohm e Leimeister (2013).

Quadro 1 - Elementos de Jogos e Motivos

Elemento de Jogo		**Motivo**
Mecânica de Jogo	**Dinâmica de Jogo**	
Documentação de comportamento	Exploração	Curiosidade Intelectual
Sistema de pontuação, medalhas e troféus	Coleção	Conquistas
Posição (*ranking*)	Competição	Reconhecimento social
Níveis e pontos de reputação	Aquisição de status	Reconhecimento social
Tarefas em grupo	Colaboração	Intercâmbio social
Pressão do tempo, tarefas e missões	Desafio	Estimulação cognitiva
Avatares, mundos virtuais, trocas virtuais	Desenvolvimento/organização	Autodeterminação

Fonte: Blohm e Leimeister (2013, p. 276)

Helms, Barneveld e Dalpiaz (2015) em seu trabalho identificaram e categorizaram os elementos de jogos utilizados em gamificação partindo de uma pesquisa na literatura. Após identificar os elementos e eliminar duplicatas, os mesmos os categorizaram com base em características em comum, evitando o máximo a ocorrência de um elemento em mais de uma categoria.

Desta forma, Helms, Barneveld e Dalpiaz (2015) identificaram sete categorias que são:

- Progressão: inclui elementos que podem ser utilizados para identificar a progressão de um jogador.

- Prêmios: essa categoria é formada por elementos que estimulem a motivação extrínseca do jogador.

- Regras: são os elementos que restringem as ações e comportamento do jogador.

- Social: todos os elementos relacionados com a interação e comunicação entre os jogadores.

- Competição: elementos que criam competição e conflito entre os envolvidos.

- Comunicação: esta categoria é composta por elementos que estejam relacionados com a comunicação entre o jogador e o jogo.

- Geral: os elementos que não se encaixam nas outras categorias foram categorizados como gerais.

Dentro destas categorias, Helms, Barneveld e Dalpiaz (2015) dividiram os elementos elencados na literatura conforme o Quadro 2.

Quadro 2 – Elementos e categorias

Categoria	**Elemento**
Progressão	Níveis
	Buscas (*Quest*)
	Enredo
	Objetivo
	Descoberta
	Solução de problema
	Personagens
	Curiosidade
Prêmios	Pontos

	Medalhas
	Recursos
	Estado de vitória
Regras	Regras gerais
	Limites de tempo
	Chance
Social	Amizade
	Fantasia
	Avatares
	Gráfico social
Competição	Quadro de posições (Leaderboard)
	Competição
	Luta contra chefes
	Desafio
Comunicação	*Feedback*
	Interação
Geral	Controle
	Diversão
	Jogar

Fonte: Adaptado de Helms, Barneveld e Dalpiaz (2015, p. 6)

Robson et al. (2015) por sua vez dividem os elementos de jogos em basicamente três tipos, os elementos de configuração, os elementos de regra e os elementos de progressão.

As mecânicas de configuração são aquelas que moldam o ambiente, definem quais são os artefatos que são necessários e como os usuários (jogadores) se estruturam para a experiência. Já as mecânicas de regra definem o objetivo e as restrições que serão aplicadas ao processo “gamificado”, podem ser utilizadas para limitar tempo ou especificar quais efeitos decorrem de determinada ação. E, por fim, as mecânicas de progressão que tem como objetivo manter a experiência em execução, estas mecânicas estão relacionadas ao *feedback* e recompensas que os usuários recebem durante o processo (ROBSON et al., 2015).

Os elementos mecânicos são escolhidos por quem estiver desenhando o processo (ou jogo) e são definidos antes de iniciar a experiência, estes elementos se mantem constantes durante o processo (ROBSON et al., 2015).

Os elementos dinâmicos, como dito anteriormente, são os comportamentos gerados ou reforçados nos participantes da experiência, estes estão relacionados a forma como os jogadores interagem com os elementos mecânicos definidos. Entretanto, estes elementos não são previsíveis, ao se planejar o processo, a escolha dos elementos mecânicos é definida com base em um provável elemento dinâmico resultante, mas o resultado continua dependente de como os usuários irão reagir, e mesmo esta reação pode não ser uniforme. Devido a isto, durante o desenho do processo deve-se tentar identificar o maior número possível de comportamentos que possam resultar de cada decisão tomada e estar preparado para cada uma delas (ROBSON et al., 2015).

Os elementos podem ser aplicados com o objetivo de atingir dois tipos de gamificação. A gamificação estrutural, onde os elementos são acrescentados ao processo com o objetivo de induzir o usuário a executá-lo, sem alterar o processo em si. E a gamificação de conteúdo, que tem por objetivo alterar o processo de maneira que o mesmo se torne mais próximo da vivência de um jogo (KAPP, 2014).

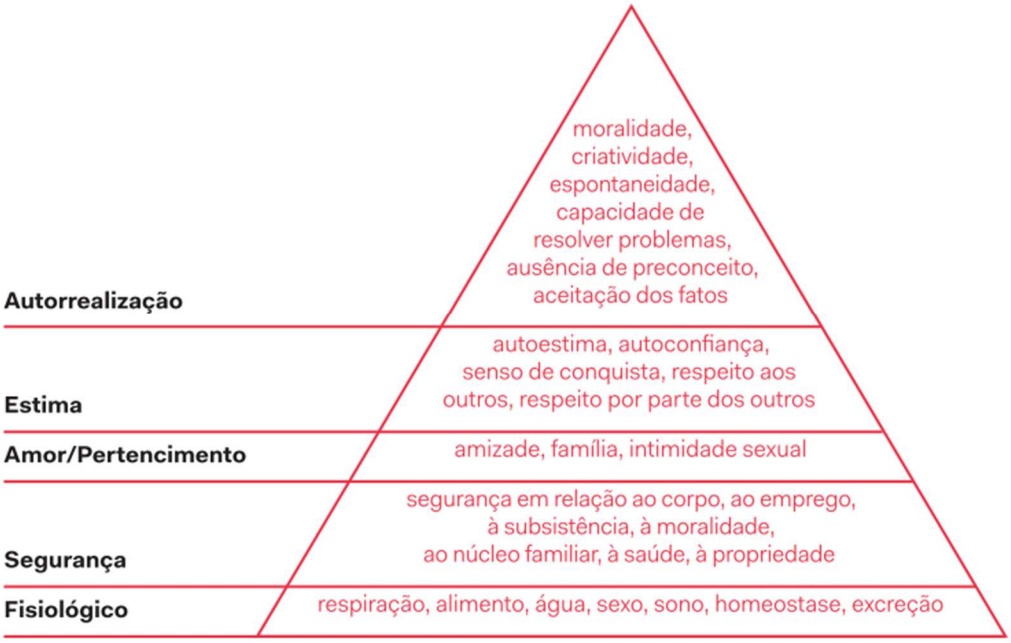
Exemplos de elementos utilizados na gamificação estrutural são os de progressão e de prêmio como os pontos, níveis e medalhas. Já na gamificação de conteúdo os elementos de progressão como enredo e curiosidade são exemplos de elementos que podem ser utilizados para aproximar o processo de um jogo (KAPP, 2014).

Esses elementos quando utilizados corretamente podem alterar o comportamento das pessoas, no capitulo a seguir são apresentadas algumas teorias que explicam o porquê desta influência.

2.3.2 Motivação

Vianna et al. (2013) argumenta que para que se possa compreender o motivo pelo qual técnicas utilizadas na gamificação influenciam o comportamento das pessoas é importante relacioná-las à Teoria da Hierarquia de Necessidades de Abraham Maslow que agrupa as necessidades humanas em forma de uma pirâmide (Figura 3).

Figura 3 – Pirâmide de Maslow



Fonte: Vianna *et al.* (2013, p. 16)

Os jogos estão diretamente relacionados com o topo da pirâmide, a autorrealização. Elementos como regras, metas definidas, *feedback* e recompensas, que são facilmente encontrados em jogos mas não em um ambiente de trabalho, conduzem o ser humano ao sentimento de satisfação quando os objetivos são atingidos (VIANNA et al., 2013).

Já a teoria da autodeterminação (*Self-determination theory*, SDT) proposta por Ryan e Deci (2000) identifica fatores que possam afetar a motivação, tanto positivamente como negativamente. A SDT define que o ser humano possui três necessidades psicológicas inatas que quando satisfeitas garantem o seu crescimento saudável e sua vitalidade (DECI; RYAN, 2000). Estas necessidades são competência, relacionamento e autonomia. Sendo que competência está relacionada com a necessidade de controlar resultados e a maestria, o relacionamento refere-se ao desejo de interagir e estar conectado a algo, e por fim, a autonomia é a necessidade de controlar sua própria vida (DICHEV et al., 2015).

Segundo Dichev et al. (2015), a principal força para se alterar o comportamento de um indivíduo é a motivação. Para cada ação executada existem diversos fatores que a induzem, entretanto é necessário algum tipo de motivador relacionado. Para Miller, Deci e Ryan (1988) a motivação é uma importante

preocupação para toda empresa que tenha como objetivo prosperar, é fundamental motivar seus funcionários. Este motivador pode ser tanto intrínseco como extrínseco.

As motivações extrínsecas estão relacionadas ao desejo de receber algo em troca de determinada ação, podendo ser algo físico, como um prêmio ou mesmo dinheiro, ou algo mais subjetivo, como o reconhecimento de outras pessoas. Mas está sempre relacionado a algo externo ao indivíduo (RYAN; DECI, 2000).

A motivação intrínseca por sua vez, é mais profunda, relacionada ao desejo do indivíduo em executar tal ação, seja por percebê-la interessante ou desafiadora. É um elemento crítico porque está relacionado diretamente com o desejo inerente que um indivíduo possui em aumentar seu conhecimento e suas habilidades (RYAN; DECI, 2000). Ainda segundo Ryan e Deci (2000), apesar da motivação intrínseca estar relacionada com o indivíduo em si, a mesma existe na relação entre as pessoas e a tarefa a ser executada. Não existe tarefa pela qual qualquer indivíduo estará intrinsecamente motivado.

Estudos tem apontado que recompensas tangíveis, bem como prazos, regras, competições e tudo que possa ser percebido como uma forma de controle acabam tendo um efeito negativo na motivação intrínseca do indivíduo. Já ações que deem liberdade ao indivíduo e controle da situação incrementam a motivação intrínseca (RYAN; DECI, 2000).

A empresa Bunchball, que se apresenta como a líder de mercado como implantadora de gamificação em empresas, considera cinco motivadores intrínsecos como os que apresentam maior impacto na motivação de um funcionário, eles estão descritos no Quadro 3 (BUNCHBALL, INC., 2014).

Quadro 3 – Cinco principais motivadores intrínsecos

Motivador	**Descrição**
Autonomia	Permitir ao funcionário fazer o que ele quer no momento em que quer desde que o trabalho seja concluído.
Maestria	Se aperfeiçoar em determinadas tarefas pode ser motivador por diversos motivos. Pode significar um trabalho mais fácil de ser realizado ou uma recompensa melhor maior pelos resultados, por exemplo.
Propósito	As pessoas gostam de sentir que seu trabalho está fazendo a diferença e que o mesmo tenha significado para a empresa.

Progresso	Pessoas se motivam quando percebem evolução em algo que se importam, seja na vida pessoal ou profissional.
Interação Social	Os seres humanos são criaturas sociais, precisam interagir e compartilhar. Também se motivam com o reconhecimento.

Fonte: Elaborado pelo Autor

Apesar da importância da motivação intrínseca, a maior parte das atividades realizadas por um indivíduo na sociedade atual não são realizadas por motivação intrínseca e sim por algum motivador externo (RYAN; DECI, 2000).

Na SDT essa motivação, mesmo que extrínseca, pode ter um certo grau de autonomia. Ryan e Deci (2000) apresentam como exemplo a motivação de dois estudantes, onde um estuda para não sofrer alguma punição por parte de seus pais, desta forma a sua motivação não está relacionada com o ganho gerado pela atividade em si, mas sim com evitar uma ação externa. Já o segundo estudante estuda por identificar neste ato a possibilidade de uma vida de sucesso no futuro. No caso do segundo estudante se identifica um sentimento de escolha e apesar de ser uma motivação extrínseca seu grau de autonomia é muito maior que o primeiro estudante. No primeiro caso a pressão é efetuada por um agente externo, já no segundo caso a pressão é interna do próprio indivíduo.

Devido a existência dessas diferenças entre casos de motivação extrínseca, Ryan e Deci (2000) dividiram a mesma em quatro tipos: regulação externa, introjeção, identificação e integração.

A motivação extrínseca chamada de regulação externa ocorre quando o comportamento do indivíduo é regulado por uma clara ação externa ou uma recompensa externa.

A introjeção ocorre quando a regulação é interna, porém as ações são tomadas devido a um sentimento de pressão, ou com o objetivo de melhorar o próprio ego ou devido ao orgulho. De acordo com Ryan e Deci (2000), apesar de ser uma regulação interna de uma pessoa, os comportamentos introjetados não são considerados como parte autônoma do ser e por isso são considerados externos (ainda que de forma menos intensa que a regulação externa).

A regulação por identificação, por sua vez, ocorre quando o indivíduo se identifica com a importância de determinado comportamento de tal forma que aceita

sua regras e as tem como regras próprias para si. Este tipo de motivação é mais autônomo e interno que os listados anteriormente.

E por fim, a forma mais autônoma de motivação extrínseca indentificada por Ryan e Deci (2000), a regulação integrada que ocorre quando o comportamento é completamente assimilado pelo indivíduo, de forma que a razão para determinada ação seja internalizada até se tornar uma autodeterminada. Esse tipo de motivação se aproxima muito da motivação intrínseca, entrentanto ainda é executado com o objetivo de alcançar algum resultado por tal comportamento.

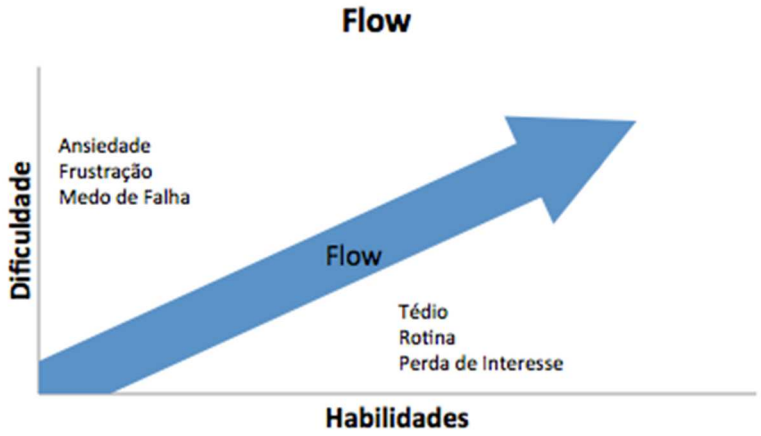
Ryan e Deci (2000) ainda entendem que um determinado comportamento pode passar por estas etapas, trocando conforme a percepção do indivíduo. Mas ao mesmo tempo, não existe uma sequência entre estes tipos. Por exemplo, um comportamento pode ser inicialmente tomado devido a uma motivação integrada, mas com o tempo se tornar uma motivação de regulação externa.

Outra teoria que é associada com o engajamento de uma pessoa com determinada tarefa é conhecida como *flow*. O estado de *flow* ocorre quando uma pessoa está extremamente focada em determinada atividade que a executa de uma forma que não percebe o passar do tempo, a fadiga e qualquer outra coisa que não seja a atividade em si (CSIKSZENTMIHALYI; NAKAMURA; ABUHAMDEH, 2005).

O estado de *flow* pode acontecer em qualquer atividade que um indivíduo esteja executando, mas geralmente ocorre em atividades onde se tem um conjunto de objetivos claros (CSIKSZENTMIHALYI, 1997) e que a dificuldade e as habilidades necessárias para executá-las sejam de certa forma balanceadas, ou seja, não podem ser simples de forma que sejam banais nem tão difíceis que acabem frustrando ou causando ansiedade no individuo (DICHEV et al., 2015).

A dificuldade deve ir aumentando conforme as habilidades do indivíduo crescem, isso para evitar o tédio. A Figura 4 apresenta o balanceamento que deve existir entre as dificuldades e as habilidades, caso isso não ocorra, pode levar o indivíduo a perda de interesse caso suas habilidades sejam muito superiores a dificuldade apresentada pelas tarefas ou a frustração caso a situação seja a inversa.

Figura 4 – *Flow* entre tédio e frustração



Fonte: Adaptado de Dichev et al. (2015)

Jogos são um exemplo de atividade que podem conduzir o jogador a um estado de *flow* pois os mesmos apresentam um conjunto claro de objetivos e um nível ideal de *feedback* e além disso podem apresentar uma dificuldade variável que pode se adaptar ao nível necessário para as habilidades do jogador (NAH et al., 2014).

Esse mesmo cenário pode ser encontrado no ambiente de trabalho, pois as principais características que são encontradas em um jogo que conduzem o jogador ao estado de *flow* podem muitas vezes aparecerer durante um dia de trabalho. Em geral as tarefas possuem um objetivo e um conjunto de regras claras, o funcionário tem um *feedback* se a atividade foi executada corretamente ou não e a dificuldade para executar a tarefa na maior parte das vezes está em um nível que o funcionário pode gerenciar (CSIKSZENTMIHALYI, 1997).

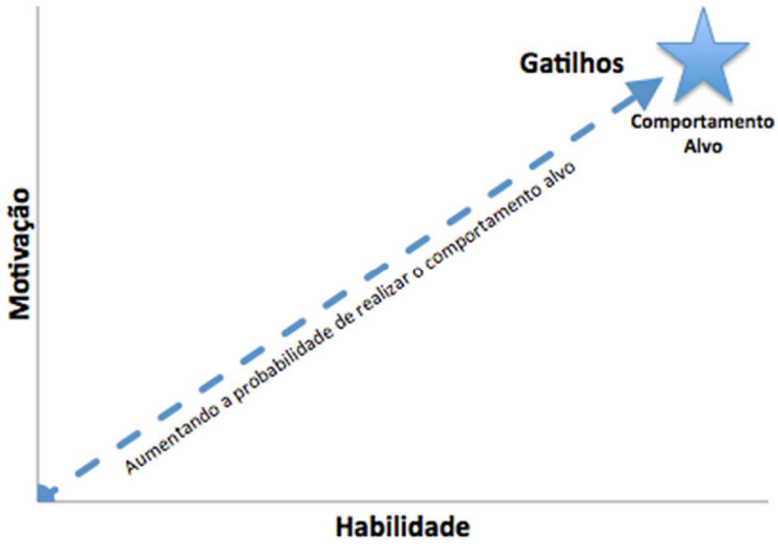
Conforme Csikszentmihalyi, Nakamura e Abuhamdeh (2005), quando o indivíduo se encontra em estado de *flow* ele se sente no controle da situação e a preocupação com o sucesso ou não acaba reduzindo, o que por sua vez é um dos motivos para que a experiência se torne agradável e até recompensadora.

Além do flow, Fogg apresenta um outro modelo para se entender o comportamento humano. De acordo com esse modelo existem três fatores que levam uma pessoa a ter determinado comportamento: motivação; habilidade e gatilhos (FOGG, 2009).

De acordo com Fogg (2009) para que um indivíduo execute um determinado comportamento ele precisa estar suficientemente motivado, deve ter a habilidade de

executar o comportamento e ser acionado para executar o comportamento. A Figura 5 apresenta uma visualização do modelo e seus três elementos.

Figura 5 – Elementos do modelo comportamental de Fogg



Fonte: Adaptado de Fogg (2009)

O comportamento alvo poderia estar em qualquer parte do gráfico e pode variar conforme o indivíduo. O modelo sugere que nem sempre aumentar a motivação é o melhor caminho, muitas vezes facilitar a execução terá o efeito desejado. Se a execução é muito simples, ela pode ser executada mesmo que a pessoa não esteja motivada, e o contrário também é verdadeiro e muitas vezes o ser humano está tão motivado que é capaz de aumentar sua habilidade para conseguir executar tal comportamento (FOGG, 2009).

O gatilho também é um elemento importante para a execução do comportamento, existem situações em que o indivíduo está motivado e possui a habilidade mas não se comporta da maneira desejada simplesmente por não possuir um gatilho para tal (FOGG, 2009).

Apenas a existência do gatilho também pode não ser suficiente, muitas vezes o momento em que o mesmo dispara é que faz a diferença necessária para a execução do comportamento (FOGG, 2009). O gatilho só é eficaz se disparar após o indivíduo atingir um certo ponto de motivação e habilidade, se disparar antes pode acabar causando frustração (DICHEV et al., 2015).

Dichev et al. (2015) descreve três tipos básicos de gatilho:

- Fagulha: é um gatilho que possui internamente um elemento motivacional. “Exemplos de fagulhas podem variar de um texto que destaca o medo até vídeos que inspiram esperança” (FOGG, 2009, p. 6).

- Facilitador: este gatilho informa ao indivíduo que executar determinado comportamento é simples e que o mesmo pode ser executado por ele. Este gatilho é muito efetivo para pessoas que possuam motivação, mas não a habilidade necessária para a realização do comportamento (FOGG, 2009).

- Sinalização: o objetivo deste gatilho é apenas o de lembrar o indivíduo de que deve realizar o comportamento. É um gatilho eficaz para pessoas que possuam motivação e a habilidade necessária para executar o comportamento (FOGG, 2009).

A possibilidade de afetar a motivação e o engajamento das pessoas faz com que a gamificação se torne uma ferramenta bastante útil em diversos cenários, o capítulo a seguir apresenta algumas aplicações onde gamificação foi utilizada para atingir os objetivos propostos.

2.3.3 Aplicações

A gamificação pode ser utilizada em diversos contextos, alguns exemplos apresentados por Alves (2014) são:

- Threadless: empresa de camisetas que utilizou uma abordagem de gamificação para acelerar a entrada da mesma no mercado.

- Bradesco: utilizou gamificação para lançar um novo produto no mercado, lançando inclusive um jogo interativo no YouTube que foi jogado por 133.435.960 segundos.

- Buffalo Wild Wings: restaurante e bar que utilizou uma campanha semelhante a uma gincana com o intuito de alcançar um maior engajamento dos clientes.

Focando em empresas de TI, Machado et al. (2015) apresenta um estudo de caso onde foi utilizado um processo de gamificação de conteúdo com o objetivo de capacitar os colaboradores do time de vendas e atendimento a clientes e parceiros em soluções de *cloud computing*.

Para tal foi criado um cenário lúdico baseado no filme O Poderoso Chefão, onde cada colaborador representava um personagem que participava de uma família e possuía um papel com direitos e deveres específicos.

Os próprios participantes definiram as famílias e seus papéis no jogo e este foi dividido em três fases. A primeira fase foi o lançamento, onde um dos organizadores apresentou as regras e o propósito do jogo de forma lúdica, inclusive caracterizado de gangster da família Corleone. A segunda fase foi a atividade em si, onde era apresentado um estudo de caso real de uma solução de *cloud computing*, compartilhando conhecimento com os participantes. E, por fim, a terceira fase que era o desafio final, onde apenas as duas famílias com mais pontos participaram de um dia de desafios e debates, e por fim uma caça ao tesouro. Ao final dessa fase foi apresentado o vencedor, chamado de *Grand Father*.

Na análise realizada por Machado et al. (2015, p.1023), evidenciou-se a presença do estado de *flow*, principalmente "[...] o estabelecimento de metas claras, concentração e imersão do participante no desenvolvimento das tarefas, sentimento de controle de consciência e de atitudes, feedbacks e equilíbrio entre as habilidades do sujeito e os desafios da atividade proposta."

Patary (2014) apresentou um estudo de caso onde a gamificação estrutural foi utilizada exatamente em um time de desenvolvimento que já trabalhava em um processo ágil.

Para gamificar o processo foram utilizados quatro passos, definir os objetivos de negócio que se deseja aperfeiçoar, definir quais habilidades, comportamentos e atitudes do jogador que se deseja, identificar as motivações, preferências, interesses e barreiras de cada jogador e iterativamente desenhar, construir e testar o jogo comportamental em conjunto com os jogadores.

Para manter os jogadores atualizados com a situação de cada um no jogo foi montado um quadro de liderança que além de apresentar os líderes, apresenta os atributos que se deseja aperfeiçoar, o nível atual de cada jogador e os *badges* que possuem. Desta forma cada um poderia acompanhar sua evolução e os pontos em cada atributo.

Para este processo também foram definidas algumas formas de celebrar o resultado, a cada quadrimestre a equipe se reunia para celebrar os maiores pontuadores e a cada seis meses um almoço era realizado para celebrar com o vencedor. Algumas vezes prêmios eram distribuídos.

Os resultados identificados por Patary (2014) após a gamificação do processo foram:

- Maior frequência de versões de demonstração e efetividade dos mesmos.

- Incrementado a velocidade do time.
- Melhoria na identificação de falhas no software em estágios iniciais.
- Aperfeiçoamento dos componentes reutilizáveis.
- Redução do lead time.
- Aumentou a participação do cliente no processo ágil.

Para criar cada um destes processos gamificados se utilizaram estratégias que foram seguidas para se definir o desenho final da solução. No capítulo seguinte será apresentado um *framework* para o desenho de sistemas gamificados que se divide em seis passos.

2.3.4 *Framework* para desenho de um sistema gamificado

Na literatura é possível encontrar diversos *frameworks* para se desenvolver um sistema gamificado. Neste trabalho de pesquisa será utilizado o processo de seis passos definidos por Werbach e Hunter (2012):

1) Definir os objetivos de negócio: Este passo é crítico para se ter um sistema gamificado bem desenvolvido, é importante identificar quais objetivos se deseja atingir com o processo gamificado (WERBACH; HUNTER, 2012).

2) Delinear os comportamentos desejados: É fundamental determinar quais são os comportamentos que os jogadores devem ter para que se atinja os objetivos de negócio definidos no passo anterior. Além de definir os comportamentos, também deve-se definir uma métrica com qual se possa medir se determinado comportamento está sendo executado satisfatoriamente.

3) Descrever os jogadores: Nesta etapa os jogadores e suas relações devem ser identificados. Também é importante identificar o que motiva cada jogador.

4) Elaborar o ciclo de atividades: Ciclos de atividade são um método muito utilizado para se modelar as ações em um sistema gamificado. Uma ação do usuário gera alguma outra atividade, esta outra atividade acaba gerando novas ações do usuário. Werbach e Hunter (2012) definem dois tipos de ciclos que podem ser utilizados para se desenhar o sistema, os laços de engajamento, onde a ação do usuário gera um *feedback* do sistema, este *feedback* por sua vez motiva o usuário a tomar novas ações. E os laços de progressão, que representam as alterações no jogo conforme o jogador vai vencendo etapas ou alcançando novos níveis.

5) Não esquecer a diversão: A diversão é importante pois ela que faz com que o jogador queira continuar no processo. Werbach e Hunter (2012) argumentam que o jogo deve conter diversos tipos de diversão, desde desafios complexos até diversão puramente casual e simples, isso porque cada jogador pode gostar de um tipo diferente de diversão.

6) Implantar as ferramentas corretas: Esta é a última etapa do *framework*, após cada decisão tomada nos passos anteriores é chegado o momento de escolher as técnicas e elementos mecânicos e dinâmicos que serão utilizadas na implementação. Este processo deve ser iterativo, você adiciona um elemento ao processo e testa o resultado do mesmo, e assim repetidamente, um verdadeiro ciclo PDCA.

3 METODOLOGIA

Neste capítulo serão apresentados os procedimentos que foram utilizados para a execução da pesquisa proposta.

Segundo Gil (2010, p. 1) a pesquisa pode ser definida "[...] como o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos".

3.1 DELINEAMENTO DA PESQUISA

Este trabalho foi realizado utilizando uma pesquisa qualitativa de nível exploratório baseada em um estudo de caso único.

A pesquisa exploratória tem como objetivo proporcionar uma maior familiaridade com o problema, com a finalidade de torná-lo mais claro e explícito ou com o objetivo de construir hipóteses (GIL, 2010). Outra característica da pesquisa exploratória é de ser um pesquisa bastante flexível e poder ser utilizada em pequenas amostras (MALHOTRA, 2011).

Ainda segundo Malhotra (2011, p. 59),

> A pesquisa exploratória pode ser usada para qualquer uma das seguintes finalidades:
> - Formular um problema ou defini-lo com mais precisão.
> - Identificar cursos alternativos de ação.
> - Desenvolver hipóteses.
> - Isolar variáveis e relações-chave para exame posterior.
> - Obter informações para desenvolver uma abordagem ao problema.
> - Estabelecer prioridades para pesquisas posteriores.

Segundo Dalfovo, Lana e Silveira (2008) a pesquisa qualitativa tem como principal material de trabalho os dados qualitativos, ou seja, os dados coletados não são expressos em números, ou quando expressos, os mesmos possuem um papel menor para a análise.

Para estudos em administração de empresas a pesquisa qualitativa pode ser associada a coleta e análise de texto e ainda com a observação direta de comportamentos (DALFOVO; LANA; SILVEIRA, 2008).

O estudo de caso, por sua vez, permite ao pesquisador identificar e reter características significativas de eventos da vida real, como, por exemplo, o comportamento de pequenos grupos e os processos organizacionais (YIN, 2010).

Com base nesta análise e no tema proposto para esta pesquisa, o estudo de caso mostrou ser a estratégia mais adequada para ser utilizada na elaboração da mesma.

A finalidade desta pesquisa é avaliar a possibilidade do uso de uma estratégia de gamificação em empresas de desenvolvimento de pequeno porte com o intuito de reduzir os retrabalhos causados por erros gerados no processo de desenvolvimento de software.

3.2 DEFINIÇÃO DA UNIDADE DE ANÁLISE

O propósito da pesquisa é avaliar a viabilidade do uso de gamificação em uma equipe de desenvolvimento de software de pequeno porte. Para a realização dessa pesquisa o objeto de estudo será a empresa de software SystemHaus S.A.

A SystemHaus é uma empresa fundada em Novo Hamburgo que iniciou a sua história como uma provedora de soluções informatizadas para diversos ramos de atuação. Por estar localizada em uma região onde o mercado do calçado e seus componentes eram bastante fortes muitos clientes eram relacionados a esta indústria.

Com o passar do tempo, a empresa identificou um nicho de mercado promissor na indústria do couro, com processos complexos não atendidos pelos *softwares* até então disponíveis, tornando-se especialista nas demandas deste segmento.

Atualmente, a empresa comercializa um produto chamado Antara, que atende curtumes de diversos portes e em diversos países, e devido a esta gama de clientes heterogênea, apesar do foco em um mercado bastante específico, o produto precisa ser altamente adaptável e flexível.

A solução tem como principal módulo o Industrial, isso porque é neste módulo que a ferramenta se diferencia das opções disponíveis no mercado. Além de possuir um conhecimento já embarcado para o mercado coureiro, a proposta é adaptar o *software* para atender a realidade do cliente.

O processo de implantação do Antara ocorre sempre no formato de um projeto e segue as seguintes fases:

Inicialização: Nesta fase ocorre as negociações iniciais e a abertura do projeto em si.

Diagnóstico: Etapa onde consultores especializados mapeiam os processos do cliente e verificam a aderência dos mesmos aos processos atendidos pelo Antara. Nesta etapa é feita a análise das lacunas e o levantamento dos requisitos das alterações. Nesta fase também são identificadas a necessidade de integrações com outros sistemas.

Desenvolvimento: Os documentos gerados pela fase anterior relacionados as alterações necessárias e desenvolvimento de integrações são as entradas desta etapa. Nesta fase o desenvolvimento destes requisitos é realizado e testado em conjunto com os implantadores e consultores.

Configuração, parametrização e preparação: Os consultores efetuam as configurações do Antara e executam os testes de aceitação. Todas as configurações são preparadas para atender as necessidades do cliente.

Treinamento: Os usuários são capacitados nesta etapa, apresentando como cada processo será realizado no sistema na prática.

Encerramento do projeto: Após a implantação é executado uma etapa de acompanhamento e por fim, com a aprovação do cliente, o projeto é concluído.

O processo de desenvolvimento na empresa SystemHaus pode ser dividido em quatro cenários diferentes, o primeiro cenário ocorre quando existe um projeto de implantação (seguindo as etapas listadas anteriormente) e são levantadas alterações, adaptações e novas funcionalidades para atender o novo cliente.

O segundo cenário ocorre quando um cliente com o sistema já implantado requisita uma alteração ou nova funcionalidade (chamado de solicitação de mudança) que não seja grande o suficiente para que se crie um novo projeto.

Quando as alterações são melhorias ou mudanças solicitadas internamente tem-se o terceiro cenário de desenvolvimento. Este pode ocorrer devido a requisitos não-funcionais, como por exemplo, melhor estrutura de instalação do software, ou por requisitos funcionais, como processos ainda não atendidos pela solução e que podem ser estratégicas para uma futura negociação.

Por fim, existe o cenário de correções de problemas (*bugs*) que o software apresenta durante o uso, porém o problema não foi encontrado durante o andamento de um projeto.

A equipe responsável por atender o desenvolvimento destes cenários é a mesma e é bastante enxuta, tendo atualmente 7 desenvolvedores e 1 estagiário. Neste ambiente o retrabalho acaba sendo um grande risco para os projetos quando

se apresenta em quantidade elevada pois acabam gerando uma grande quantidade de trabalho não previsto e que muitas vezes não pode ser acomodado no cronograma ou extrapola o custo do projeto.

Esta equipe utiliza *Scrum* para atender projetos maiores e *Kanban*[1] para o atendimento de chamados de manutenção e solicitações de mudança. O foco do trabalho é a utilização de gamificação associado ao ambiente *Scrum*, pois este é um *framework* já bastante difundido em empresas de *software* e desta forma esta pesquisa pode ter um valor maior para o mercado.

Além disso o *Scrum*, por sua natureza de trabalhar com *timeboxes*, já apresenta um dos elementos que são utilizados em gamificação como forma de pressão que é o limite de tempo.

3.3 TÉCNICA DE COLETA DE DADOS

Para a realização deste trabalho foram utilizadas três técnicas de coleta de dados.

A pesquisa bibliográfica, que foi utilizada para gerar todo o material do referencial teórico. Esta pesquisa foi realizada em artigos, livros e materiais de empresas que trabalham com gamificação com o intuito de fornecer uma base ao pesquisador. Segundo Gil (2010), a pesquisa bibliográfica é um requisito para praticamente qualquer pesquisa, seja para fundamentar a pesquisa ou para permitir que o pesquisador consiga investigar uma gama muito maior de fenômenos sem que o mesmo tenha que realizar a pesquisa diretamente.

A segunda técnica de coleta é a pesquisa em registros de arquivo, onde serão investigados os registros mantidos pela empresa, principalmente as informações de retrabalho. Estas informações são controladas por um *software* proprietário chamado *Compass*, onde toda a tarefa realizada pelos setores da empresa devem ser registrados. Esses tipos de registros foram avaliados pelo pesquisador para garantir a exatidão dos mesmos e a condição em que os mesmos foram produzidos (YIN, 2010).

A ferramenta *Compass* controla o fluxo do processo e dispõe de informações de retrabalho que foram fundamentais para esta pesquisa. A análise foi feita com

[1] *Kanban* refere-se a um método ágil de desenvolvimento de *software* criado por David Anderson com base no quadro kanban utilizado no Sistema Toyota de Produção.

base nos registros realizados durante o período de 01/10/2015 até 15/01/2016, com isso foram considerados um total de 665 chamados (cada requisição realizada na ferramenta é um chamado) nesta pesquisa.

A terceira técnica de coleta utilizada foram entrevistas com os participantes do processo que esse trabalho pretendia estudar. Estas entrevistas foram realizadas utilizando a técnica chamada de grupo focal.

Essa técnica consiste em entrevistas em grupo, sendo que uma pessoa executa o papel de moderador (neste caso o pesquisador) para conduzir o grupo pelos tópicos desejados pelo entrevistador. Toda a informação dita pelo grupo durante uma sessão de discussão é um dado essencial para a pesquisa (MORGAN, 1997). É um método de grande utilidade para explorar ideias, pensamentos, atitudes e experiências dos envolvidos em relação a um determinado assunto, a discussão gerada pode encorajar pessoas a justificar uma determinada ideia e por isso tem um grande potencial de obter mais informações do que em uma entrevista individual (PLUMMER-D'AMATO, 2008).

O grupo focal foi utilizado em dois momentos da pesquisa, em um primeiro um grupo formado por cinco indivíduos mais o moderador/pesquisador para discutir sobre as causas de retrabalho que poderiam ser evitadas no processo de desenvolvimento (retrabalhos corretivos). Esse grupo foi formado por seis desenvolvedores, todos com pelo menos um ano de empresa. A duração dessa entrevista foi de aproximadamente uma hora e o Apêndice A contém as questões que serviram de roteiro para a entrevista.

No segundo momento a entrevista foi voltada para elencar ideias com o objetivo de preparar um processo gamificado, os entrevistados e o moderador foram os mesmos. Esta entrevista foi montada a partir de uma análise realizada sobre a primeira sessão, bem como com os dados extraídos dos registros do *Compass* e da revisão bibliográfica. O roteiro da entrevista pode ser encontrado no Apêndice B.

Após a realização destas entrevistas o pesquisador reuniu os dados coletados e elaborou uma proposta de processo gamificado. Essa proposta foi enviada via correio eletrônico a todos os participantes do grupo focal para que os mesmos pudessem avaliar o resultado e indicar se acreditavam que o processo teria o efeito desejado no índice de retrabalho da equipe.

3.4 TÉCNICAS DE ANÁLISE DE DADOS

Nesta pesquisa foram levantadas informações de três fontes distintas, conforme descrito anteriormente, portanto, para a análise e validação dos dados coletados foi utilizada a triangulação destas informações e a relação das mesmas com os objetivos propostos nesta pesquisa. Segundo Gil (2010), a triangulação deve ser usada confrontando informações extraídas das diversas fontes de pesquisa para garantir uma maior credibilidade e segurança ao pesquisador.

3.5 LIMITAÇÕES DO MÉTODO

Por se tratar de um estudo de caso único e de apenas uma empresa os resultados encontrados não podem ser generalizados, seria necessário mais estudos para garantir a generalização.

Outra dificuldade encontrada é que gamificação ainda é um assunto relativamente novo e a quantidade de estudos e materiais sobre o mesmo é relativamente limitada. Muitos dos materiais científicos disponíveis seguem a mesma linha, principalmente quando relacionado a aplicações e usos, o que acabou limitando o pleno atingimento do segundo objetivo deste trabalho.

4 APRESENTAÇÃO E ANÁLISE DOS DADOS

Este capítulo tem como objetivo apresentar os dados coletados nos registros da empresa (*software Compass*) e as informações extraídas das entrevistas realizadas, além de relacioná-los ao referencial teórico e aos objetivos da pesquisa.

4.1 REGISTRO DE ARQUIVOS

Os registros de arquivos foram coletados com dois objetivos principais, identificar o fluxo atual do desenvolvimento e coletar informações sobre retrabalho.

Na empresa estudada esses itens são controlados pela ferramenta *Compass*, essa ferramenta controla todo o fluxo de trabalho da empresa. Sendo que para toda solicitação, seja ela uma nova demanda ou uma falha encontrada, é aberto um chamado no *Compass*, o qual irá controlar todo o ciclo de vida da solicitação dentro da empresa. Neste chamado, podem ser anexados arquivos e observações sobre o problema ou solução, além disso mantém a rastreabilidade das pessoas que trabalharam no chamado e o tempo gasto por cada uma delas, e no caso de um desenvolvimento, ainda registra os arquivos alterados no código fonte do *software*.

O *Compass* possui o conceito de filas, onde cada setor pode criar suas filas para controlar o chamado dentro de seu processo interno. Como este trabalho tem como foco o processo dentro do desenvolvimento, ele trata apenas das filas relativas a este setor. Para a entrada de um chamado no setor de desenvolvimento existem duas filas principais. A fila Desenvolvimento – Suporte, que é a fila onde as demandas de correções ou alterações pequenas iniciam sua vida dentro do setor. E a fila Desenvolvimento – Projeto, por onde entram as demandas relativas a um projeto.

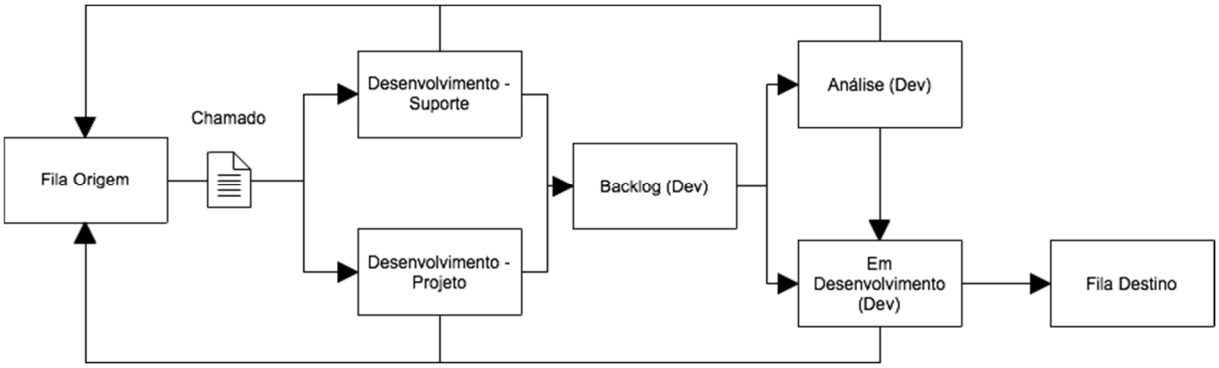
Após a entrada do chamado no desenvolvimento, o mesmo é avaliado conforme sua prioridade (no caso de projeto, utiliza-se o *Scrum* com o *Product Owner* definindo o valor de cada demanda e a equipe definindo o que fará parte do *Sprint* durante o *Sprint Planning*. Nos outros casos o gestor da área trabalha sincronizado com o setor inicial da demanda que deve definir a prioridade) e os chamados são transferidos de fila para *Backlog* (Dev). No caso do *Scrum* todas as demandas selecionadas para o *Sprint* são transferidas para o *Backlog* (Dev), no caso de manutenção e alterações não relacionadas a um projeto, os chamados vão

sendo transferidos para manter uma média de 6 chamados na fila de *backlog*, isso garante que caso algum chamado novo urgente chegue na fila de desenvolvimento, o mesmo não demore para ser enviado para *backlog*.

Após a priorização dos chamados, os mesmos ficam disponíveis para a equipe. Novamente a demanda troca de fila, caso seja possível atender o chamado com as informações contidas nele, o chamado passa para a fila Em Desenvolvimento (Dev) e fica associado com o desenvolvedor responsável (aquele que selecionou o chamado). Caso uma análise técnica mais aprofundada seja necessária, o chamado passa pela fila Análise (Dev) também vinculado ao desenvolvedor responsável (neste caso após a análise realizada, o chamado fica novamente disponível para qualquer desenvolvedor).

A Figura 6 apresenta este processo, percebe-se também que em algumas filas o chamado pode retornar para a fila origem, isso porque existem casos em que o chamado não pode ser realizado (por não ser uma falha, pois é necessário passar por orçamento antes de desenvolver, etc.) ou porque o chamado não está completo e é necessário mais informação para a execução do mesmo.

Figura 6 – Ciclo de vida de um chamado no desenvolvimento



Fonte: Elaborado pelo autor

Sempre que a fila é trocada uma nova observação é vinculada a essa demanda. Essa observação é automática quando a transferência não requisita nenhuma informação extra, porém sempre que é a etapa final do desenvolvimento essa observação conterá informações importantes sobre a solução realizada e se essa solução pode ser considerada um retrabalho. Caso seja, o desenvolvedor precisa identificar o tipo de retrabalho. Os tipos de retrabalho são:

- Código incorreto: Este tipo de retrabalho deve ser apontado quando o retrabalho foi causado por algum erro no código, que provavelmente não tenha sido testado pelo desenvolvedor e passou para a fase seguinte.

- Mal entendimento: Neste caso o desenvolvedor não compreendeu corretamente o requisito da solicitação e a solução ficou incorreta.

- Mal documentado: Existe uma linha muito tênue entre o mal entendido, o mal documentado e o cenário não especificado. O mal documentado seria quando a documentação do requisito está incorreta ou incompleta de forma que induza o desenvolvedor a não entregar o que realmente irá ter valor para o cliente.

- Cenário não especificado: O cenário não especificado é um tipo específico de documentação incompleta, neste caso a funcionalidade desenvolvida entrega o que foi requisitado, mas algum cenário não estava especificado na documentação.

- Versão incorreta: Este retrabalho pode ocorrer na empresa, pois a mesma possui diversas versões do produto rodando atualmente e este retrabalho ocorre quando a correção é realizada na versão incorreta.

- Retrabalho gerado por outro chamado: Neste caso uma nova solicitação é realizada para atender um problema causado por outro chamado, esta solicitação pode ter basicamente dois motivos, a alteração feita pelo chamado anterior é incorreta e deve ser modificada ou a alteração acabou adicionando problemas ou falhas em um cenário que já era atendido corretamente.

Se comparar estes retrabalhos com os tipos de retrabalhos apresentados por Fairley e Willshire (2005), percebe-se que todos eles podem ser categorizados como retrabalhos evitáveis, principalmente do tipo corretivo. Outro ponto importante é que neste caso os retrabalhos evolutivos não são destacados na empresa, isso porque esse número não é de interesse da mesma, uma vez que grande parte dos chamados serão exatamente retrabalhos evolutivos ou customizações para um cliente específico.

Como o foco desta pesquisa é adicionar elementos de gamificação no processo de desenvolvimento para reduzir os retrabalhos causados por erros que poderiam ser evitados no próprio desenvolvimento, os retrabalhos que serão contabilizados são os relativos a código incorreto, mal entendimento e retrabalho gerado por outro chamado.

Um ponto importante a se ressaltar é que uma demanda pode inicialmente não ser um retrabalho, mas após a primeira entrega, o chamado pode retornar para o desenvolvimento e neste momento se tornar um retrabalho. Por isso a informação mais relevante serão as observações de saída da fila Em Desenvolvimento (Dev).

Conforme descrito no capítulo 3, os dados analisados compreendem chamados realizados no período de 01/10/2015 a 15/01/2016, totalizando 665 chamados. Estes 665 chamados passaram pela fila Em Desenvolvimento (Dev) 920 vezes, estes retornos nem sempre estão relacionados a um retrabalho, algumas vezes pode ser apenas uma requisição de mais informações. O Quadro 4 apresenta um resumo dos dados levantados a partir do software *Compass*.

Quadro 4 – Quantidade de retrabalhos no setor desenvolvimento

	Outubro 2015	**Novembro 2015**	**Dezembro 2015**	**Janeiro 2016**	**Total**	**% Retrabalho**
Chamados	150	209	236	70	665	-
Ocorrências	201	292	332	95	920	-
Retrabalhos	72	93	119	40	324	35,22
Código Incorreto	36	48	61	17	162	17,61
Mal Entendimento	11	14	13	10	48	5,22
Mal Documentado	5	4	8	2	19	2,07
Cenário não especificado	11	11	19	6	47	5,11
Versão Incorreta	1	2	3	1	7	0,76
Retrabalho gerado por outro chamado	8	14	15	4	41	4,46

Fonte: Elaborado pelo Autor

No Quadro 4 é apresentado as quantidades de ocorrências, ou seja, o número de vezes que o chamado passou pelo desenvolvimento, e também o percentual de ocorrência de retrabalhos total e separado pelos tipos apontados. Por esse percentual se identifica que grande parte dos retrabalhos podem ser evitados no desenvolvimento, principalmente no caso de código incorreto onde a fonte do retrabalho é o próprio desenvolvimento.

No capítulo seguinte estas informações de retrabalho retiradas do software são analisadas com base nos dados coletados nas entrevistas.

4.2 GRUPO FOCAL - RETRABALHO

O primeiro grupo focal realizado teve como principal objetivo extrair informações sobre como os desenvolvedores percebem o retrabalho, os tipos

identificados, as principais causas e possíveis melhorias no processo para reduzir a taxa de retrabalho.

As perguntas 1 e 2 do questionário (apêndice A) tiveram como principal objetivo iniciar o assunto e validar se a importância que o retrabalho tem no desenvolvimento de software esta claro entre os desenvolvedores, bem como os diferentes tipos de retrabalho existentes.

Praticamente todos os envolvidos disseram saber que o retrabalho afeta negativamente um projeto, inclusive apresentando exemplos de casos que ocorreram. Se mostraram apreensivos inclusive com casos onde o cliente é afetado por erros que tenham passado pelo desenvolvimento e muitas vezes o teste não tenha sido realizado ou realizado em um cenário diferente do real. Entretanto, na visão dos entrevistados os problemas eram muito mais relacionados a pontos anteriores ao desenvolvimento (levantamento ou documentação) do que a algo que o desenvolvimento pudesse controlar ou evitar. Entretanto ao analisar os dados apontados de retrabalho vistos no capítulo anterior, percebe-se que a realidade é que muitos retrabalhos estão relacionados exatamente ao setor de desenvolvimento.

Um dos entrevistados, inclusive, apontou a falta de uma definição clara e de cenários de teste como um dos grandes motivos para que alguns chamados tenham que ir para o teste e voltar para o desenvolvimento diversas vezes, muitas vezes tendo que reestruturar todo o trabalho pois não estava claro o objetivo da solicitação.

No momento de separar os retrabalhos por tipo, ficou claro que não se poderia contabilizar os retrabalhos evolutivos pois este é basicamente o motivo da existência do setor, uma vez que a empresa possui um produto, a maior parte das solicitações são evoluções no produto.

Desta forma restaram os retrabalhos do tipo evitável, e alguns entrevistados entendem que desta categoria uma parte considerável poderia ser evitada dentro do próprio setor.

A questão 3 está relacionada com a identificação de causas de retrabalhos, neste ponto foram listados diversos motivos, novamente com exemplos dos ocorridos.

Uma das causas discutidas foi o nível de teste realizado no próprio desenvolvimento, alguns erros simples acabam passando para a fase seguinte. Alguns dos motivos para essa ocorrência são a falta de conhecimento sobre uma determinada regra de negócio por parte do desenvolvedor; a documentação ser

muito superficial, não apontando algum caso específico; a complexidade do cenário e a falta de uma boa base para o teste com os dados necessários; e por fim, a multiplicidade de cenários (este foi um dos motivos mais aceito pelo grupo, todos se identificaram com este motivo), pois o produto desenvolvido suporta diversas configurações diferentes e é difícil de testar todas as possibilidades.

Outras causas levantadas foram a documentação ruim ou pouca informação na solicitação, a falta de uma base de dados onde se possa simular o problema ou mesmo o novo desenvolvimento, e até o fato de deixar de enviar algum arquivo para o servidor de versões por esquecimento de algum desenvolvedor.

A questão 4 teve como objetivo tentar relacionar motivação com retrabalho, já tentando avaliar gamificação como uma abordagem válida neste cenário.

Foi consenso da equipe entrevistada que o retrabalho afeta o psicológico do desenvolvedor e muitas vezes o desenvolvedor quer verificar o que fez errado. Um dos entrevistados ressaltou que sente-se mal quando um chamado feito por ele retorna como retrabalho e buscar identificar onde estava o erro e se era responsável pelo mesmo.

Entretanto, os entrevistados relataram que o fato de um retrabalho afetar o desenvolvedor, em um primeiro momento acaba fazendo com que ele se torne mais cuidadoso nos passos seguintes pois não quer repetir o erro, inclusive foi colocado como algo normal do ciclo de aprendizagem. Ou seja, a motivação nesse caso é afetada mas não tem um resultado negativo imediato, pelo contrário, acaba tornando o desenvolvedor mais cauteloso.

Já quando questionado sobre como a motivação em geral afeta o retrabalho (foi requisitado aos entrevistados que imaginassem um dia em que chegassem na empresa sem motivação para o trabalho) a maioria acredita que afeta a quantidade de retrabalho gerado e muitas vezes porque nesse caso não se consegue focar no que se está fazendo. Para um dos entrevistados o principal problema da falta de motivação está mais relacionada com o tempo que se demora para realizar determinada tarefa e não tanto com a qualidade. Já para outro, ele acredita que quando a falta de motivação está relacionada com o trabalho em si, as taxas de retrabalho tendem a subir pois neste dia o desenvolvedor não está preocupado com o resultado de suas ações. E nesse ponto da entrevista argumentaram que o resultado do efeito do retrabalho na motivação difere do efeito da baixa motivação

com o trabalho em si. A primeira não chega a ser tão negativa, ao menos em curto prazo, já a segunda sim, afeta o resultado da empresa diretamente.

Na questão número 5 o objetivo era, sabendo que retrabalhos sempre existirão, levantar melhorias para identificá-los mais cedo no processo, e desta maneira reduzir o custo dos mesmos. Nesta questão foi requisitado que não se levasse em consideração apenas o setor do desenvolvimento e sim todos os setores da empresa que participam do processo de desenvolvimento de um projeto ou manutenção. Um dos problemas levantados que dificulta a identificação do retrabalho em uma etapa inicial é o tempo entre cada fase do projeto. Nos casos em que esse tempo é maior o retrabalho se apresenta em maior quantidade e muitas vezes em momentos mais avançados do projeto.

Outra melhoria apontada que poderia facilitar a identificação de retrabalhos seria possuir uma base com cenários prontos para que os desenvolvedores pudessem simular diversas situações diferentes já na etapa de desenvolvimento. Entretanto, um dos entrevistados apontou que isso não resolve quando o próprio cliente não possui uma ideia clara de sua necessidade e o mesmo sugeriu a utilização de prototipação como solução para evitar esse tipo de retrabalho, essa inclusive é uma técnica apontada por Boehm e Papaccio (1988) como responsável por diminuir retrabalhos.

O que se percebeu é que das ideias levantadas para minimizar identificação dos retrabalhos em etapas tardias muitas ações acabam não dependendo apenas do setor de desenvolvimentos, e sim, que muitas vezes envolve outro setores.

Já na questão 6 se buscou identificar melhorias no processo de desenvolvimento com o objetivo de reduzir os retrabalhos em si. A primeira ideia levantada foi aproximar a equipe responsável por determinado projeto, essa aproximação seria através da realocação dos colaboradores de forma que estivessem posicionados lado a lado. Desta forma, em um projeto *scrum* o time estaria realmente próximo, permitindo a troca de informações de forma dinâmica e imediata.

Um dos entrevistados propôs o uso de treinamentos ministrados pelos próprios desenvolvedores como forma de fixar técnicas novas e ferramental utilizado no desenvolvimento e desta forma evitar retrabalhos causados por falta de domínio destas. Esse tipo de treinamento já é utilizado esporadicamente pela equipe,

entretanto, algo percebido pelos que participaram é que geralmente os mais beneficiados são os responsáveis por aplicar o treinamento.

Quando proposto o uso de *pair programming* por um dos entrevistados não se chegou a um consenso da validade do uso da técnica, enquanto alguns desenvolvedores defenderam o uso pois tiveram boas experiências, outros afirmaram que nas oportunidades que tiveram de utilizar a mesma não perceberam um resultado tão efetivo que valesse o uso. Por fim, concluíram que é uma técnica que pode ser utilizada esporadicamente de preferência com desenvolvedores que possuam ideias diferentes para que ambos possam ter uma maior chance de crescimento.

Outra técnica comentada foi a utilização de revisão de código para evitar que códigos com erros passem para a etapa seguinte, entretanto também foi uma técnica questionada, principalmente pelo fato de poder criar um ambiente negativo e conflitante entre o codificador e o revisor. Inclusive um dos desenvolvedores trouxe um exemplo vivido por ele em outra empresa onde o ambiente muitas vezes não era o ideal, simplesmente por este conflito entre revisor e desenvolvedores.

Com essa questão foi concluída a entrevista relacionada ao retrabalho. O que se percebeu nas respostas é que a gamificação pode ser utilizada de duas formas distintas, primeiro com uma abordagem motivacional, visto que foi identificado que a motivação e o foco possuem um papel importante na quantidade de retrabalhos gerados pela equipe de desenvolvimento. A outra abordagem seria utilizar gamificação para facilitar a adoção de novas tarefas que foram identificadas como tendo um provável efeito positivo no controle de retrabalhos.

O próximo capítulo apresenta o resultado do grupo focal com o objetivo de discutir soluções de gamificação para a redução da taxa de retrabalho.

4.3 GRUPO FOCAL – GAMIFICAÇÃO

Este grupo focal teve como objetivo identificar os jogadores e como estes percebem a questão motivacional por trás do processo de gamificação. No *framework* definido por Werbach e Hunter (2012) esta é uma etapa importante para uma implementação de sucesso da gamificação.

Apesar de ter um roteiro de perguntas (Apêndice B) para conduzir esta entrevista, o objetivo principal era de gerar discussão entre os participantes e extrair dessa discussão os detalhes relevantes para este trabalho.

Em uma etapa anterior ao início da entrevista, o pesquisador promoveu uma reunião informal de alinhamento com os entrevistados sobre o que é gamificação, seus elementos e uma breve explicação da teoria comportamental por traz desta técnica.

A primeira pergunta buscou identificar quais tipos de tarefa do dia a dia dos desenvolvedores são mais motivadoras. Um dos entrevistados relatou que o que mais lhe motiva é a possibilidade de criar algo novo ao invés de corrigir problema ou efetuar alterações em códigos já existentes, os outros concordaram porém analisaram que uma tarefa que seja desafiadora é ainda mais motivadora. Entretanto, um dos desenvolvedores apontou que um dos problemas do desafio é quando este é tão difícil que acaba se tornando desmotivador. No referencial teórico sobre o estado de *flow*, Dichev et al. (2015) identificou exatamente isso, onde a tarefa se apresenta difícil de tal forma que cause frustração e ansiedade.

Nesta mesma pergunta os entrevistados concluíram que um incentivo monetário não tem um efeito motivador tão efetivo, algo também identificado no referencial teórico sobre SDT onde Ryan e Deci (2000) afirmam que a motivação intrínseca é muito mais profunda se comparada aos motivadores externos.

Todos concordaram que tarefas repetitivas, por sua vez, tem um efeito negativo na motivação e que muitas vezes não se percebe a evolução quando a tarefa se repete constantemente e isso acaba levando o desenvolvedor a querer concluir esta tarefa o mais rápido possível para poder se livrar da mesma, levando ao descuido que acaba gerando retrabalho.

Por fim, um dos entrevistados ressaltou a importância de se ter um objetivo claro e atingível para que o desenvolvedor não se desmotive.

Se percebeu aqui que alguns dos motivadores listados estão relacionados com alguns dos elementos de gamificação, Hamari, Koivisto e Sarsa (2014) apresentam os elementos como desafio, objetivos claros e *feedback* como sendo alguns dos principais encontrados em sua revisão bibliográfica.

As questões 2 e 3 tiveram como objetivo verificar a validade do *feedback* imediato e o reconhecimento social como motivadores para a sequência de tarefas do desenvolvedor. Todas as questões seguinte também possuem o objetivo de

identificar ciclos de atividade que possam ser utilizados na proposta de gamificação conforme definido no *framework* apresentado por Werbach e Hunter (2012).

Um dos entrevistados tem a opinião que o *feedback* pode ser motivador quando o mesmo for positivo e quando o desenvolvedor estar envolvido com a tarefa executada. Entretanto, quando não existe um sentimento de vínculo com a tarefa esse *feedback* acaba não lhe motivando. Outro desenvolvedor discordou dessa afirmação pois acredita que o problema não é o *feedback* e sim a falta do vínculo do desenvolvedor com a tarefa.

Dois dos entrevistados percebem que ter um *feedback* de suas tarefas pode ser motivador, mas que atualmente só recebem um *feedback* quando a solicitação retorna devido a algum problema, ou seja, só quando o *feedback* é negativo.

A questão sobre reconhecimento social está relacionada com o motivador de interação social relatado por Bunchball Inc. (2014) e foi unânime entre os entrevistados quanto ao efeito positivo na motivação. Também foi questionado sobre o uso de *badges* para reconhecer o trabalhado dos desenvolvedores, identificando quem já atingiu algum nível de excelência em determinado assunto (também é um método de *feedback*). Entretanto foi apontado a necessidade de criar um sistema justo para que todos possam evoluir e alcançar os *badges* do sistema gamificado. Um desenvolvedor também sugeriu utilizar níveis dentro de um *badge* para que a pessoa tenha a motivação para evoluir cada vez mais aquele determinado conhecimento, essa abordagem é apresentada por Werbach e Hunter (2012) como um ciclo de atividade utilizando um laço de progressão.

A quarta questão teve o objetivo de validar a aceitação e a efetividade na opinião dos entrevistados para o uso de gamificação com o objetivo de adicionar novas técnicas ou reforçar já existentes no processo de desenvolvimento.

Na discussão, os participantes concluiram que tem valor para iniciar novas técnicas por ser mais um motivador, entretanto nem todos são afetados da mesma maneira pois vai depender de como o indivíduo encara a existência de pontos e *leaderboard* no processo. Um dos entrevistados acredita que com certeza não iria desmotivar a equipe, mas ao mesmo tempo lembrou do problema que poderia ocorrer caso a competição fosse levada tão a sério que um jogador pudesse tentar boicotar o outro.

Um dos entrevistados acredita que incoscientemente uma pontuação ou o uso de *badges* poderia levá-lo a executar tarefas que não está habituado apenas pela

motivação de poder perceber sua própria evolução. Este tipo de *feedback* foi bem aceito pelo grupo, onde cada indivíduo pode ver seu próprio crescimento e até ter uma visão de quantas solicitações está atendendo em um determinado período.

Outra preocupação levantada pelos entrevistados, durante esta discussão, foi se o rendimento dos desenvolvedores com mais experiência não iria desmotivar os com menos, pois estes não identificariam uma oportunidade de alcançar uma posição mais elevada em um *leaderboard*, por exemplo.

Por fim, a questão número 5 abordou a competição, que é um dos elementos dinâmicos utilizados na gamificação (BLOHM; LEIMEISTER, 2013) e pode ser gerado por diversos elementos mecânicos.

Um dos entrevistados argumentou que isso vai depender de como for elaborado o jogo, e que a competição poderia fazer com que um indivíduo pudesse querer monopolizar informações para não deixar que outros jogadores pudessem também adquirir esse conhecimento. A competição também foi questionada em relação a como o último colocado iria se comportar, se iria se motivar para buscar uma melhor posição ou se iria ficar desmotivado por não conseguir ultrapassar outros jogadores.

Após uma breve discussão, os entrevistados sugeriram o uso de equipes ao invés de indivíduos sendo colocados em um ranking, esta ideia teve grande aceitação. Os times poderiam ser montados de maneira mais justa, e poderiam ainda ser alterados com o objetivo de balancear a força de cada time.

No capítulo seguinte é apresentada uma análise das informações levantadas nas entrevistas e nos registros da empresa.

4.4 ANÁLISE DOS DADOS LEVANTADOS

Os dados levantados nos registros da empresa e a primeira entrevista realizada buscaram atender o primeiro objetivo deste trabalho que visava identificar as fontes de retrabalho no desenvolvimento e sua importância na totalidade de retrabalhos de uma empresa de software de pequeno porte.

Na empresa estudada a maior parcela de retrabalhos está exatamente associada a erros no código (conforme visto no capítulo 4.1). Esse dado mostra a importância de evitar erros nesta fase do desenvolvimento, principalmente porque na entrevista os próprios desenvolvedores perceberam que estes erros podem levar um

projeto ao fracasso, seja pela entrega fora da data do cronograma, seja pelo custo fora do previsto.

Outra informação importante extraída na entrevista é de que a falta de motivação é um fator importante para a geração de retrabalho, pois é um dos aspectos que a gamificação busca aperfeiçoar. Além disso, levantaram-se algumas ações que poderiam também reduzir a quantidade de retrabalho.

Com base nessas informações pode-se supor que a gamificação poderia ser eficaz se utilizada para aumentar a motivação dos desenvolvedores e para introduzir novas atividades ao processo com o intuito de reduzir os erros de software.

Essa suposição conduziu a pesquisa à segunda entrevista realizada. Nesta entrevista buscou-se opiniões sobre quais tipos de abordagem de gamificação poderiam ser eficazes no cenário estudado. A meta desta entrevista era servir de fonte para se atender o terceiro objetivo desta pesquisa, que é propor uma abordagem gamificada no processo de desenvolvimento de software.

Analisando-se a resposta dos entrevistados percebeu-se um certo nível de incerteza sobre a eficiência da gamificação para redução dos retrabalhos, apesar de concordarem que os benefícios listados na literatura deveriam sim reduzir este índice. Com base nisso, concluiu-se que a proposta deveria ser cuidadosamente montada, analisando cada discussão listada no capítulo 4.3, especialmente quanto ao assunto competição e cooperação.

Essa proposta também levou em consideração as aplicações de gamificação encontradas na literatura, porém não se encontrou muitas aplicações no cenário que a pesquisa propôs. Desta maneira o segundo objetivo da pesquisa que envolvia analisar aplicações no desenvolvimento e manutenção de software se resumiu a poucos exemplos.

4.5 PROPOSTA DE GAMIFICAÇÃO

Neste capítulo serão apresentados os elementos escolhidos para montar esta proposta de gamificação. Para a escolha dos elementos foram considerados os tipos de retrabalho e sugestões extraídos das entrevistas e da pesquisa bibliográfica.

Estes elementos foram adicionados ao *framework scrum*, que por si só já adiciona elementos que combinam a uma abordagem gamificada e proporciona

autonomia e objetivos claros que Csikszentmihalyi (1997) afirma serem fundamentais para se alcançar o estado de *flow*.

Para o desenho da solução gamificada foi utilizado o *framework* de Werbach e Hunter (2012) apresentado no capítulo 2.3.4. A proposta é apresentada nos subtópicos que seguem.

4.5.1 Definir os objetivos de negócio

O primeiro passo do *framework* é definir quais os objetivos de negócio que se pretende atingir com a gamificação. Essa proposta tem como principal objetivo reduzir o índice de retrabalhos corretivos no desenvolvimento de software em pequenas equipes *scrum*. O foco são os retrabalhos que podem ser evitados pela equipe de desenvolvimento.

Com base no que foi identificado na bibliografia e nas entrevistas, a gamificação pode ser utilizada para introduzir novas práticas ao processo e motivar os desenvolvedores para que sejam mais focados em suas atividades.

4.5.2 Delinear comportamentos desejados

Para atingir o objetivo definido é necessário identificar os comportamentos que se deseja produzir nos participantes do processo gamificado, este é o segundo passo do *framework*. Para essa definição se utilizou as informações da pesquisa bibliográfica e das entrevistas realizadas. A lista de comportamentos desejados é apresentada no Quadro 5, bem como as métricas para avaliar o comportamento.

Quadro 5 – Comportamentos e Métricas

Comportamento	**Métrica**
Reforçar o uso de testes unitários	Índice de teste unitários em relação a quantidade de retrabalho nos chamados testados
Maior troca de informações entre os desenvolvedores	Índice de sessões de Pair Programing executadas em relação a quantidade de retrabalho gerada nos chamados trabalhados com uso dessa técnica
Incentivar refatoração para facilitar alterações (MARTIN, 2008) e até mesmo encontrar falhas (FOWLER, 2014)	Índice de sessões de refatoração por quantidade de retrabalho nas classes refatoradas

Incentivar treinamentos internos	Índice de treinamentos aplicados na prática em relação ao treinamentos executados
Incentivar time de desenvolvedores a questionar o *product owner* por cenários de teste	Índice de Cenários de testes documentados em relação a quantidade de retrabalhos gerados em chamados com cenários de teste

Fonte: elaborado pelo autor

4.5.3 Descrever os jogadores

A terceira etapa do *framework* busca identificar os jogadores no processo gamificado. Para atender esta etapa foram utilizadas as informações levantadas nas duas entrevistas.

Os jogadores-alvo deste processo gamificado será o time de desenvolvedores da empresa analisada.

O principal motivador identificado durante as entrevistas é o desafio, todos os participantes afirmaram se motivar quando enfrentam um problema complexo. Entretanto, para que o comportamento desejado seja executado pelo desenvolvedor, o desafio precisa ser algo que aquele determinado indivíduo tenha capacidade de resolver com suas experiências e capacidades técnicas atuais, caso contrario a motivação não será suficiente para a execução da tarefa (FOGG, 2009). A dificuldade dos desafios deve aumentar de maneira proporcional a evolução do participante, seguindo um ciclo de progressão conforme definido por Werbach e Hunter (2012) em seu *framework*.

Outro item identificado como motivacional é a novidade, ou seja, tarefas com técnicas, ferramentas e tecnologia novas ou mesmo tarefas que necessitem que uma pedaço de *software* seja criado a partir do zero.

Será evitado aquilo que desmotiva os participantes, e um item identificado nas entrevistas como desmotivador foi o fato de que muitas solicitações não são claras a ponto do desenvolvedor compreender a fundo o objetivo de tal solicitação. Além de desmotivar, esse tipo de chamado deixa margem para o mal entendimento, que por sua vez leva ao retrabalho, e deve ser evitado.

4.5.4 Elaborar o ciclo de atividades

Esta etapa do *framework* define como será o andamento do processo gamificado.

A primeira etapa servirá de introdução ao jogo (Werbach e Hunter (2012) definem essa etapa como *onboarding*), que tem como objetivo apresentar como o jogo funciona para os jogadores. Na verdade, essa fase será de um *sprint* do *scrum*, onde os jogadores conseguirão atingir os primeiros níveis de forma bastante rápida, conseguindo pontos e recebendo os primeiros *badges* para compreenderem o funcionamento do processo.

Para que o jogo tenha um certo nível de competição, mas para que isso não tenha o objetivo de comparar indivíduos, o time do *scrum* será dividido em duas equipes e os pontos serão contabilizados sempre para a equipe. Desta forma, a proposta abrange o uso da competição, mas ao mesmo tempo incentiva a cooperação pois o membros do time poderão se auxiliar. Ainda com o objetivo de cooperação, *badges* especiais serão distribuídos para aqueles que cooperarem inclusive com a outra equipe, essa ação está associada a necessidade de relacionamento identificada por Deci e Ryan (2000) na teoria de autodeterminação. Este tipo de ação também resultará em uma quantidade extra de pontos. Os *badges* serão individuais e também geram pontos extras para a equipe.

Durante o planejamento do *sprint* as equipes trabalharão juntas para identificar as tarefas a serem executadas e definir o peso de cada uma. Entretanto, um novo artefato será gerado neste evento, além do *backlog* do *sprint*, será necessário distribuir as tarefas entre as equipes gerando uma lista de tarefas distintas para cada equipe. Para que a distribuição seja justa, no primeiro *sprint* será sorteado qual equipe terá o direito de escolher a primeira tarefa, as outras tarefas serão escolhidas alterando as equipes. Cada tarefa irá gerar um número de pontos de acordo com sua complexidade, desta forma cada equipe pode montar estratégias diferentes para montar a lista de tarefas que irá atender. Cada equipe deverá definir sua velocidade para que atenda as tarefas conforme esse limitador, ou seja, a equipe não pode se sobrecarregar com tarefas mais complexas. Isto permite que ambos os times possam montar seus próprios caminhos e está relacionado com uma das necessidades apresentada pela SDT, a autonomia (DECI; RYAN, 2000).

Caso uma equipe selecione uma tarefa e não consiga atendê-la, os pontos da mesma serão contabilizados para a outra equipe, e caso a tarefa não seja aceita pelo *product owner* a outra equipe receberá os pontos em dobro.

A cada tarefa concluída o jogador receberá o *feedback* imediato e seus pontos serão contabilizados para a equipe. No final, durante o *sprint review* novos pontos serão distribuídos por equipe que tiver suas tarefas avaliadas e indicadas como concluídas pelo *product owner*. Caso as duas equipes consigam concluir suas tarefas e o *sprint* for considerado um sucesso, novos pontos serão distribuídos para cada equipe, desta forma é importante que ambas equipes consigam atingir seus objetivos.

Algumas tarefas especiais, como adicionar testes unitários e refatorar um código existente gerarão pontos para a equipe e *badges* para o desenvolvedor. Desta maneira o jogo incentivará esse tipo de atitude nos jogadores, pois não são tarefas obrigatórias para se concluir o *sprint* mas são muito valorizadas.

Haverá momentos entre os *sprints* que serão chamados de *Tempo de Maestria*, que serão treinamentos onde cada equipe pode sugerir um determinado assunto que pretende aprender e ensinar aos outros participantes do processo. A escolha dependerá do alinhamento do assunto com o que a empresa almeja e uma avaliação dos próprios jogadores. O instrutor receberá *badges* ou níveis pelo treinamento quando um desenvolvedor utilizar o aprendizado na prática, a equipe do desenvolvedor que utilizou esse conhecimento também receberá pontos.

Ainda para motivar os participantes, quando uma equipe atingir determinado nível, os mesmos terão o direito de atender um desafio (em geral protótipos utilizando tecnologias novas) que funcionará como uma *boss fighting* de um vídeo game. No final do desafio a produção da equipe será avaliada por um comitê (formado pelas pessoas que serão beneficiadas pelo protótipo e que será definido previamente) que irá determinar quantos pontos a equipe receberá pelo resultado.

Os ciclos de progressão funcionarão através da evolução dos *badges* a nível individual e com *rewards* a nível de equipe, os *rewards* serão pequenos prêmios simbólicos para representar o nível da equipe.

4.5.5 Não esquecer a diversão

Alguns dos elementos apresentados nos ciclos de atividade foram adicionados com o intuito de proporcionar um certo nível de diversão aos participantes. A competição entre equipes, por exemplo, teve principalmente esse objetivo.

Outro elemento adicionado para buscar a motivação intrínseca dos desenvolvedores foi o *boss fighting*, que foi baseado nos elementos motivadores levantados nas entrevistas, especialmente desafio e inovação.

4.5.6 Implantar as ferramentas corretas

Para atender o processo gamificado proposto foram apresentados diversos elementos, estes elementos são sumarizados a seguir.

A tríade PBL, pontos, *badges* e *leaderboard* serão os principais elementos da proposta do processo gamificado. Os três elementos serão utilizados principalmente como uma forma de *feedback* imediato. Os pontos e o *leadboard* permitirão o acompanhamento e feedback sobre o desempenho dos comportamentos dos times e os badges de cada indíviduo. Os *badges* serão divididos em níveis, também sendo utilizados como elemento de progressão. O *leaderboard* e os pontos também serão utilizados como um recurso para a competição.

Pontuações e *badges* serão utilizados para incentivar o compartilhamento entre os envolvidos no processo em diversos momentos, em forma de treinamento, *pair programming* e como incentivo ao trabalho em equipe (mesmo de times diferentes).

A divisão de tempo dos *sprints* associado com as desvantagens de não concluir uma tarefa serviriam de um elemento de pressão de tempo para gerar o elemento dinâmico desafio. Outro elemento utilizado para gerar desafio no processo foi o *boss fighting*.

Para que se controle todos estes elementos, inicialmente pode ser utilizado uma simples planilha e, posteriormente, uma ferramenta mais adequada poderia ser implementada para contabilizar os pontos e gerenciar os eventos do processo (como por exemplo as sessões de refatoração e *pair programming*).

No caso da empresa estudada, a ferramenta Compass poderia ser evoluída para adicionar essas funcionalidades, pois a mesma já possui grande parte das informações atualmente, necessitando apenas contabilizar os pontos.

4.6 AVALIAÇÃO DA PROPOSTA

Com o objetivo de avaliar a proposta apresentada no capítulo anterior, foi enviado para os participantes das duas entrevistas um resumo das regras com as pontuações de cada ação no jogo (conforme o apêndice C). Além do resumo foi solicitado que os mesmos respondessem uma pergunta bem simples: Você acredita que esse processo gamificado teria sucesso quando aplicado com o intuito de reduzir os retrabalhos corretivos no setor de desenvolvimento?

Além desta pergunta, o entrevistado foi incentivado a adicionar a resposta qualquer comentário sobre o processo, ideia ou crítica.

Nas respostas recebidas percebeu-se que a maior parte dos entrevistados acredita que a redução do retrabalho pode ser alcançada com a aplicação de tal proposta, seja por meio da motivação que a equipe terá para garantir a aceitação do *product owner*, ou seja pelo nivelamento que as trocas de informações incentivadas na proposta geram.

Entretanto, um dos entrevistados disse que, para poder avaliar melhor o efeito da proposta no índice de retrabalhos, o ideal seria criar um projeto piloto para que, desta maneira, fosse possível uma avaliação real desta abordagem.

Também foi sugerido, uma vez que um dos comportamentos incentivados é a criação de testes unitários, que caso um time faça com que um teste unitário pare de ter sucesso na sua execução, contabiliza-se pontos para o outro time. E também receba pontos o time que corrigir um teste que estava em estado de erro.

Outro ponto levantado por um dos entrevistados é que esse processo gamificado só teria sucesso se todos os indivíduos participassem do mesmo de maneira voluntária, para que não o boicotassem, mesmo que inconscientemente.

5 CONSIDERAÇÕES FINAIS

A importância da redução de retrabalho em uma empresa de desenvolvimento de *software* foi identificada em diversos trabalhos na revisão bibliográfica. O custo do retrabalho pode representar em alguns casos até a maior parte do custo total de um projeto de desenvolvimento de *software*. Por este motivo este trabalhou buscou apresentar uma proposta que tivesse como objetivo reduzir este custo.

A gamificação é uma técnica que tem como objetivo motivar indivíduos para que se comportem da maneira desejada pelo designer do processo utilizando elementos que normalmente são encontrados em jogos. A proposta aqui apresentada utiliza estes elementos para atuar sobre algumas das causas de retrabalhos identificadas tanto na revisão bibliográfica quanto nas entrevistas realizadas.

O primeiro objetivo foi atingido através da pesquisa bibliográfica, onde se levantou diversas fontes de retrabalho em um ambiente de desenvolvimento de *software*. Além disso, por meio da análise dos dados do software da empresa e das entrevistas verificou-se as fontes na empresa estudada e se relacionou as mesmas com as identificadas na bibliografia.

O segundo objetivo não foi atingido plenamente pois não foram encontrados muitas aplicações no contexto desejado. Por esse motivo se avaliou aplicações de outros contextos para se compreender o uso dos elementos de gamificação na prática. A falta deste tipo de estudo destaca a contribuição deste trabalho e enfatiza a importância de novos estudos neste campo.

A proposta de gamificação para atingir o terceiro objetivo especifico desta pesquisa foi idealizada com base nas entrevistas e nas informações da revisão bibliográfica. Após a análise destas fontes se identificou que a gamificação poderia ser utilizada para minimizar algumas das causas de retrabalho identificadas nesta pesquisa e se utilizou os elementos estudados para montar a proposta.

Pode-se afirmar que, baseado no que foi apresentado na pesquisa, a mesma atingiu seu objetivo principal, pois foi feito a análise do uso de gamificação em um ambiente de desenvolvimento de *software* de pequeno porte, e concluiu-se que a proposta inicial avaliada tem potencial para reduzir o número de retrabalhos. Entretanto, seria importante passar por um processo de refinamento, conforme

identificado no último passo do *framework* de Werbach e Hunter (2012) e pelas próprias respostas de alguns dos entrevistados que avaliaram a proposta.

Por esse motivo, como proposta para trabalho futuro sugere-se o refinamento da proposta gamificada na prática, aplicando a mesma em uma empresa de *software* de pequeno porte no formato de um projeto piloto e iterativamente ir adicionando novos elementos ao processo e ajustando os existentes.

Outra sugestão para trabalho futuro é estudar os resultados alcançados no retrabalho na prática, aplicando a proposta no processo de desenvolvimento *scrum* do time.

REFERÊNCIAS

AFSHARIAN, Sharareh; GIACOMOBONO, Marco; INVERARDI, Paola. A framework for software project estimation based on cosmic, dsm and rework characterization. In: **Proceedings of the 1st international workshop on Business impact of process improvements**. ACM, 2008. p. 15-24.

ALVES, Flora. **Gamification: como criar experiências de aprendizagem engajadoras : um guia completo do conceito à prática**. 1. ed. São Paulo: DVS Editora, 2014.

ATKINSON, Roger. Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. **International journal of project management**, v. 17, n. 6, p. 337-342, 1999.

BLOHM, Ivo; LEIMEISTER, Jan M. Design of IT-based enhancing services for motivational support and behavioral change. **Business & Information Systems Engineering**, p. 275-278, 2013.

BOEHM, Barry W.; BASILI, Victor R. Top 10 list [software development]. **Computer**, v. 34, n. 1, p. 135-137, 2001.

BOEHM, Barry W.; PAPACCIO, Phillip N. Understanding and controlling software costs. **IEEE Transactions on Software Engineering**, v. 14, n. 10, p. 1462–1477, 1988.

BUNCHBALL INC. **Using Gamification to Engage Employees**, 2014. Disponível em: <http://www.bunchball.com/resources/using-gamification-to-engage-employees>. Acesso em: 15 nov. 2016.

CASS, Aaron G.; OSTERWEIL, Leon J.; WISE, Alexander. A Pattern for Modeling Rework in Software Development Processes. **Trustworthy Software Development Processes**, n. Springer Berlin Heidelberg, p. 305–316, 2009.

CHUA, Bee Bee. Rework requirement changes in software maintenance. **Proceedings - 5th International Conference on Software Engineering Advances, ICSEA 2010**, p. 252–258, 2010.

CONROY, Patrick; KRUCHTEN, Philippe. Performance norms: An approach to rework reduction in software development. **Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on**, p. 1–6, 2012.

CSIKSZENTMIHALYI, Mihaly. Finding Flow. **Psychology Today**. Julho, p. 46–71, 1997.

CSIKSZENTMIHALYI, Mihaly.; ABUHAMDEH, Sami; NAKAMURA, Jeanne. Flow A General Context. **Handbook of Competence and Motivation**, p. 598–608, 2005.

DALFOVO, Michael Samir; LANA, Rogério Adilson; SILVEIRA, Amélia. Métodos quantitativos e qualitativos: um resgate teórico. **Revista Interdisciplinar Científica Aplicada**, v. 2, n. 3, p. 1-13, 2008.

DAWSON, Christian; DAWSON, Ray. Software Development Process Models: A Technique for Evaluation and Decision-Making. **Knowledge and Process Management**, v. 21, n. 1, p. 42-53, 2014.

DECI, Edward L.; RYAN, Richard M. The “What” and “Why” of goal pursuits: Human needs and the self-determination of behavior. **Psychological Enquiry**, v. 11, n. 4, p. 227–268, 2000.

DESPA, Mihai Liviu. Comparative study on software development methodologies. **Database Systems Journal**, v. 5, n. 3, p. 37–56, 2014.

DETERDING, Sebastian et al. From game design elements to gamefulness. **Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments - MindTrek ’11**, p. 9–11, 2011.

DETERDING, Sebastian et al. Gamification. using game-design elements in non-gaming contexts. In: **CHI'11 Extended Abstracts on Human Factors in Computing Systems**. ACM, 2011. p. 2425-2428.

DICHEV, Christo et al. From Gamification to Gameful Design and Gameful Experience in Learning. **Cybernetics and Information Technologies**, v. 14, n. 4, p. 80–100, 2015.

DUVERNET, Amy M.; POPP, Eric. Gamification of workplace practices. **TIP: The Industrial-Organizational Psychologist**, v. 52, p. 39-44, 2014.

FAIRLEY, Richard E.; WILLSHIRE, Mary Jane. Iterative rework: The good, the bad, and the ugly. **Computer**, v. 38, n. 9, p. 34–41, 2005.

FOGG, Brian J. A behavior model for persuasive design. **International Conference on Persuasive Technology (Persuasive)**, p. 40, 2009.

FOWLER, Martin. **Refatoração: Aperfeiçoando o Projeto de Código Existente**. 1. ed. Porto Alegre: Bookman, 2014.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 5. ed. São Paulo: Atlas, 2010.

HAMARI, Juho. **Definig Gamification**. Disponível em:

<http://juhohamari.com>. Acesso em: 30 ago. 2015.

HAMARI, Juho; KOIVISTO, Jonna; SARSA, Harri. Does gamification work? - A literature review of empirical studies on gamification. **Proceedings of the Annual Hawaii International Conference on System Sciences**, p. 3025–3034, 2014.

HELMS, Remko W.; BARNEVELD, Rick; DALPIAZ, Fabiano. **A Method for the Design of Gamified Trainings. PACIS 2015 Proceedings,** 2015.

HUMPHREY, Watts S. **The Personal Software Process (PSP)**. Software Engineering Institute, Carnegie Mellon University. Novembro, 2000.

HUOTARI, Kai; HAMARI, Juho. Defining gamification: a service marketing perspective. In: **Proceeding of the 16th International Academic MindTrek Conference**. ACM, 2012. p. 17-22.

JONES, Capers. Software project management practices: Failure versus success. **CrossTalk: The Journal of Defense Software Engineering**, v. 17, n. 10, p. 5-9, 2004.

KAPP, Karl M. What L&D professionals need to know about gamification. **Spring**, p. 16–19, 2014.

MACHADO, Lisiane. et al. A Gamificação como Estratégia de Capacitação e o Estado de Flow : um Estudo de Caso em uma Empresa da Área de Tecnologia da Informação (TI) da Região Sul do Brasil. **XIV Simpósio Brasileiro de Jogos e Entretenimento Digital**, p. 1015–1024, 2015.

MALHOTRA, Naresh K. **Pesquisa de Marketing: Uma Orientação Aplicada**. 6. ed. Porto Alegre: Bookman, 2011.

MARTIN, Robert C. **Clean Code: A Handbook of Agile Software Craftsmanship**. 1. ed. Upper Saddle River: Prentice Hall PTR, 2008.

MILLER, Karen A.; DECI, Edward L.; RYAN, Richard M. Intrinsic Motivation and Self-Determination in Human Behavior. **Contemporary Sociology**, v. 17, n. 2, p. 253, mar. 1988.

MORGAN, David L. **The Focus Group Guidebook**. 1. ed. Thousand Oaks: SAGE Publications, Inc., 1997.

NAH, Fiona F. et al. Flow in gaming: literature synthesis and framework development. **International Journal of Information Systems and Management**, v. 1, p. 83–124, 2014.

PATARY, Chandan L. Gamifying Agile projects to Drive Employee Engagement and Increase Performance. **PM World Journal**, v. 3, n. 11, p. 1–18,

2014.

PLUMMER-D'AMATO, Prudence. Focus group methodology part 1: considerations for design. **International Journal of Therapy & Rehabilitation**, v. 15, n. 2, p. 69–73, 2008.

RAJAMARTHANDAN, Senthil. **Using Gamification to Build a Passionate and Quality-Driven Software Development Team**. Disponível em: <http://www.cognizant.com/InsightsWhitepapers/Using-Gamification-to-Build-a-Passionate-and-Quality-Driven-Software-Development-Team.pdf>. Acesso em: 15 dez. 2016.

ROBSON, Karen et al. Is it all a game? Understanding the principles of gamification. **Business Horizons**, v. 58, n. 4, p. 411–420, 2015.

RYAN, Richard M.; DECI, Edward L. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. **Contemporary educational psychology**, v. 25, n. 1, p. 54–67, 2000.

STARK, George et al. An Examination of the Effects of Requirements Changes on Software Maintenance Releases. **Journal of Software Maintenance**, v. 11, n. 5, p. 293–310, 1999.

SUTHERLAND, Jeff; SCHWABER, Ken. **The scrum guide. The definitive guide to scrum: The rules of the game.** Disponível em: <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf>. Acesso em: 5 nov. 2015.

TOOR, Shamas-ur-Rehman; OGUNLANA, Stephen O. Beyond the "iron triangle": Stakeholder perception of key performance indicators (KPIs) for large-scale public sector development projects. **International Journal of Project Management**, v. 28, n. 3, p. 228–236, 2010.

VIANNA, Ysmar et al. **Gamification Inc.: Como reinventar empresas a partir de jogos**. 1. ed. Rio de Janeiro: MJV Press, 2013.

WERBACH, Kevin; HUNTER, Dan. **For the win: How game thinking can revolutionize your business**. 1. ed. Wharton Digital Press, 2012.

WESTLAND, J. Christopher. The cost behavior of software defects. **Decision Support Systems**, v. 37, n. 2, p. 229–238, 2004.

YIN, Robert K. **Estudo de Caso: Planejamento e métodos**. 4. ed. Porto Alegre: Bookman, 2010.

APÊNDICE A

Questões Focus Group - Retrabalho

N	Questão	Objetivo	Referência
1	Considerando retrabalho como a alteração em qualquer uma das seguintes dimensões: funcionalidade, estrutura, comportamento e atributos de qualidade. A taxa atual de retrabalho é um fator que está afetando o sucesso dos projetos? Justifique.	Esclarecer a importância do retrabalho no desenvolvimento de software.	Jones (2004)
2	Sabendo que os retrabalhos podem ser divididos em evitáveis e inevitáveis, e que os evitáveis podem ainda ser divididos em de retrospectiva e corretivos, indique quais os tipos de retrabalho mais comuns no seu ambiente.	Identificar tipos de retrabalho.	Fairley e Willshire (2005)
3	Considerando os retrabalhos evitáveis, quais a principais causas que podem ser identificadas?	Identificar causas de retrabalho	Fairley e Willshire (2005)
4	Na sua opinião essa frequência de retrabalhos afeta a motivação da equipe? De que forma? E a motivação, afeta o retrabalho?	Identificar a relação entre motivação e retrabalho.	Fairley e Willshire (2005); Conroy e Kruchten (2012)
5	O processo atual de desenvolvimento pode ser aprimorado para que os retrabalhos sejam identificados rapidamente, evitando que apareçam apenas em etapas mais avançadas do projeto? Como?	Identificar pontos de melhoria no processo	Boehm e Basili (2001)
6	O processo atual de desenvolvimento pode ser aprimorado para diminuir a quantidade de retrabalhos evitáveis? Como?	Identificar pontos de melhoria no processo	Boehm e Basili (2001)

APÊNDICE B

Questões Focus Group – Gamificação

N	Questão	Objetivo	Referência
1	Quais tipos de tarefas são mais motivadoras? Entre as tarefas elencadas, é possível colocar em ordem da mais motivadora para a menos motivadora?	Identificar os jogadores.	Werbach e Hunter (2012), Bunchball Inc. (2014)
2	O *feedback* imediato a uma tarefa motiva o desenvolvedor a seguir para a próxima tarefa? E se esse feedback se desse na forma de pontos ou *badges,* seria mais efetivo?	Definir ciclos de atividade	Werbach e Hunter (2012), Rajamarthandan (2014)
3	O reconhecimento dos colegas é um motivador importante? Uma estrutura de níveis onde qualquer um possa perceber a evolução dos outros seria benéfico?	Avaliar a importância do reconhecimento social	Werbach e Hunter (2012), Blohm e Leimeister (2013), Bunchball Inc. (2014)
4	Fazer uso de um *leaderboard* e de pontos em determinadas tarefas, como por exemplo, testes unitários, incentivaria o desenvolvedor a executar este tipo de tarefa com maior frequência? E como isso poderia ser sustentado?	Definir ciclos de atividade	Werbach e Hunter (2012)
5	Um *leaderboard* é um bom método de *feedback* mas ao mesmo tempo é um elemento que gera competição. Esse tipo de competição seria saudável para uma equipe? Se não, como torná-la saudável?	Definir elementos mecânicos para o processo gamificado	Robson et al. (2015)

APÊNDICE C

Proposta de gamificação de um processo de desenvolvimento de uma equipe *scrum* com o objetivo de redução de retrabalho.

Regras:

A equipe *scrum* deve ser dividida em dois times, um nome deve ser definido para cada time, esse nome constará no *leaderboard*.

Durante o *sprint planning* os próprios times definirão quantos pontos cada tarefa irá valer (dentro de um limite entre 1 e 20) e farão a distribuição de tarefas conforme a velocidade escolhida para cada time. Esta distribuição será feita de maneira intercalada, onde cada time terá a chance de escolher uma tarefa e em seguida o próximo time fará o mesmo processo. Caso um time selecione tarefas que completem sua velocidade primeiro, o outro time irá escolher tarefas sucessivamente até sua velocidade ser completada ou acabarem as tarefas.

É importante que a equipe saiba que uma tarefa selecionada e não concluída fará com que o time adversário receba os pontos da tarefa e caso uma tarefa considerada concluída seja negada pelo *product owner* durante o *sprint review*, a equipe adversária receberá os pontos em dobro.

Nos primeiros 1000 pontos que um time alcançar, o mesmo receberá um desafio que é um chefe de fase, esse chefe consiste em um protótipo utilizando uma nova tecnologia ou uma solução técnica para um problema que necessita ser resolvido no software. A solução para o desfio será avaliada por um comitê que será formado de acordo com o desafio e caso seja considerado que o time derrotou o chefe, receberá um pequeno prêmio.

Outro método para o time ganhar pontos é propor um treinamento, conhecido como Tempo de Maestria, o mesmo ocorrerá a cada dois *sprints* e os times podem escolher assuntos e concorrer a uma chance de apresentá-los.

A tabela abaixo indica a pontuação recebida por cada ação do time.

Ação	**Pontos**
Tarefa concluída	O número de pontos definidos durante o *sprint planning* (entre 1 e 20).
Tarefa considerada concluída pelo *product owner*	Cada tarefa irá gerar 30% extra de pontos, para isso durante o *sprint review* todos os pontos serão somados e então será aplicado os 30%, sempre arredondando para cima.

Badge recebido	Cada *badge* renderá 10 vezes o seu próprio nível em pontos.
Teste unitário	Cada classe de teste unitário renderá 10 pontos
Refatoração	Cada sessão de refatoração renderá 10 pontos
Pair programming	Cada sessão de *pair programming* renderá 10 pontos
Auxilio prestado a elementos do time adversário	30 pontos
Tempo de Maestria – treinamento aplicado	50 pontos para o instrutor e 25 pontos para o indivíduo que comprovadamente utilizar o conhecimento ministrado (o instrutor só receberá os pontos quando alguém utilizar o conhecimento).
Derrotar um chefe – *boss fighting*	100 pontos
Se ambos os times concluírem o *sprint*	50 pontos extras para cada time