Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Introdução de Gamificação no Desenvolvimento de Software

Danilo Batista Medeiros

Número de Ordem PPGCC: M013

Teresina-PI, 20 de julho de 2015

Danilo Batista Medeiros

Introdução de Gamificação no Desenvolvimento de Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Pedro de Alcântara dos Santos Neto

Teresina-PI

20 de julho de 2015

Danilo Batista Medeiros

Introdução de Gamificação no Desenvolvimento de Software

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Trabalho aprovado. Teresina-PI, 23 de julho de 2015:

Pedro de Alcântara dos Santos Neto
Orientador

Esteban Walter Gonzalez Clua
Convidado 1

Erick Baptista Passos
Convidado 2

Raimundo Santos Moura
Convidado 3

Teresina-PI
20 de julho de 2015

Resumo

O desenvolvimento de software é, em muitos momentos, uma atividade excitante e desafiadora, mas em outros pode se apresentar como uma atividade enfadonha, sendo necessário a introdução de estímulos para incentivar sua realização. A introdução de elementos de jogos em atividades como o ensino de engenharia de *software* e desenvolvimento de *websites* é algo já consolidado e que evidencia que atividades do mundo real podem incorporar elementos de *game design*, visando com isso torná-las mais interessantes e engajantes. Neste trabalho é proposto a introdução de elementos de *game design* nas atividades relacionados ao desenvolvimento de software, incluindo elementos para estimular a aderência às prescrições do processo de desenvolvimento e estimular um aumento de produtividade da equipe. Para isso, foram idealizadas diversas mecânicas de jogos, que foram avaliadas com dados retrospectivos e também avaliadas em um estudo de caso realizado em uma empresa privada. Os resultados obtidos indicam que a gamificação aplicada ao desenvolvimento de software estimulou os desenvolvedores a realizar suas tarefas diárias, indicando assim que a aplicação de técnicas de *game design* pode contribuir positivamente para a atividade.

Palavras-chaves: desenvolvimento de *software*, scrum, *game design*, gamificação.

Abstract

Software development is many times an exciting and challenging activity, but at other time it can be presented as a boring activity. The introduction of game elements in activities like software engineering learning and web sites building shows that real-world activities can incorporate elements of game design making them more interesting and fun. This paper proposes the introduction of elements of game design in software development, including elements to encourage the adherence to the process prescriptions and stimulate the productivity. We show the idealized mechanics and results obtained from a case study on a software development team in a private company. The gamification applied to software development encouraged developers to increase their productivity, indicating that this idea is useful to improve the activity.

Keywords: software development, scrum, game design, gamification.

Lista de ilustrações

Figura 1 – Organização estrutural de partes das prescrições do Scrum *framework*. 18

Figura 2 – Exemplo do gráfico *burndown*. Os eixos vertical e horizontal representam o montante de trabalho a ser realizado durante o *sprint* (*sprint backog*) e seu tempo de duração, respectivamente. 20

Figura 3 – O laço desafio-punição-recompensa: quando o jogador enfrenta um desafio, suas ações são confrontadas com as regras do jogo, que definem o resultado como uma recompensa ou uma punição. 26

Figura 4 – Grafo hierárquico de desafios. O grande desafio (Game) pode ser encarado como uma composição de subdesafios (Level), que por sua vez podem ser subdivididos em outros subdesafios (Quest). 27

Figura 5 – Insígnias de um usuário da ferramenta *RedCritter Tracker* ganhados como recompensa por realização de tarefas. 42

Figura 6 – Mapeamento dos componentes Scrum em um grafo hierárquico de desafios. 50

Figura 7 – Fragmento de tela do *redmine* com o protótipo desenvolvido mostrando apresentação de *xp* (valor 13), *level* (valor 3) e insígnias obtidas na realização de tarefas. 61

Figura 8 – Exemplo de visualização de uma tarefa premiada com uma insígnia. . . 61

Figura 9 – Exemplos de notificações apresentadas em eventos como alteração de *xp*, *level* e obtenção de insígnias. 62

Figura 10 – Fragmento da tela ilustrando a visualização de um quadro de classificação de desenvolvedores. 62

Figura 11 – Visualização de perfil de jogador. 63

Figura 12 – Comparação entre as equipes dos projetos quanto aos resultados obtidos para os *achievements* propostos. 71

Figura 13 – Gráfico *Burndow* construído como um mapa para aventuras RPG. . . . 74

Figura 14 – Quadro de pontuação dos desenvolvedores. As pontuações representam avanços de início e conclusão de tarefas por cada desenvolvedor de acordo com os pontos de história estimados por cada tarefa. 75

Figura 15 – *Implicitly Estimated Velocity* médio para períodos semanais de cada equipe participante do estudo de caso. 88

Figura 16 – *Comparativo da evolução de HoC (Hits of Code)* médio entre as dez semanas anteriores(azul) e durante(vermelho) a aplicação de gamificação para as quatro equipes e geral da empresa. 89

Lista de tabelas

Tabela 1 – Exemplo de valores da relação entre Xp *Level*. 60
Tabela 2 – Resultados individuais para o *achievement clockwork developer* agrupados por projeto: total de medalhas ganhas por cada desenvolvedor durante os *sprints* considerados nessa avaliação. 69
Tabela 3 – Resultados para o *achievement* Clockwork Team: níveis alcançados para cada projeto das equipes e o número total de *sprints* concluídos como planejado. 69
Tabela 4 – Resultados para o *achievement Sprint Backlog Completion*: níveis alcançados para cada equipe dos projetos e o total de *sprints* com o *sprint backlog* concluído. 70
Tabela 5 – Resultados para o *achievements Sprint Latency*: níveis indicando ausência de intervalo entre *sprints*. 70
Tabela 6 – Listagem das notas de percepção de produtividades extraídos antes e após a gamificação para a empresa, equipe e respectivo desenvolvedor. 85
Tabela 7 – Listagem das notas médias de percepção de produtividades por equipe extraídas antes e após a gamificação para a empresa, equipe e respectivos desenvolvedores. 86
Tabela 8 – Fatores definidos como determinísticos para os patamares de produtividade apontados pelos desenvolvedores antes da aplicação de gamificação. 86
Tabela 9 – Influências positivas ao ambiente de trabalho identificadas pelos desenvolvedores após da aplicação de gamificação. 87
Tabela 10 – Sumarização das respostas quanto a continuidade de prática de gamificação no cotidiano de desenvolvimento dos desenvolvedores participantes do estudo. 87

Lista de abreviaturas e siglas

CD	*Clockwork Developer*
CT	*Clowoerk Team*
DSTR	*Daily Scrum Developer Presente*
GM	*Game Manager*
IEV	*Implicitly Estimated Velocity*
LOC	*Lines of code*
PO	*Product Owner*
POPa	*Partner POPa*
POPR	*Presence*
RPG	*Role Playing Game*
SBC	*Sprint Backlog Completion*
SL	*Sprint Latency*
SM	*Scrum Master Alterado*
SRA	*Sprint Review Acceptance*
SSM	*Super Scrum Master*
TDD	*Desenvolvimento Orientado a Testes.*

Sumário

I INTRODUÇÃO 3
Contextualização . . . 5
Motivação . . . 6
Objetivos e contribuições . . . 7
Organização da dissertação . . . 7

II FUNDAMENTAÇÃO TEÓRICA 9

1 DIVERSÃO E MOTIVAÇÃO . . . 13
1.1 Diversão . . . 13
1.1.1 Diversão e jogos . . . 14
1.2 Motivação . . . 14

2 SCRUM . . . 17
2.1 Estrutura . . . 17
2.2 Papéis, cerimônias e artefatos . . . 18

3 PRODUTIVIDADE . . . 21
3.1 Definição . . . 21
3.2 Métricas . . . 21

4 CONCEITOS DE *GAME DESIGN* . . . 25
4.1 Mecânicas . . . 25
4.2 Um *framework* de jogos genérico . . . 26
4.3 *Achievements* . . . 28
4.4 *Feedback* imediato . . . 29
4.5 Trapaça . . . 30
4.6 RPG - *Role Playing Games* . . . 30

5 GAMIFICAÇÃO . . . 33
5.1 Definição . . . 33
5.2 Casos e exemplos de aplicação . . . 33
5.3 O outro lado da gamificação . . . 35

6 ESTUDOS DE CASO . . . 37
6.1 Definição . . . 37
6.2 Estudos de caso e a engenharia de software . . . 38

7 TRABALHOS RELACIONADOS 41
7.1 Gamificação 41
7.2 Jogos sérios e aprendizagem 41
7.3 Gamificação em ferramentas 42
7.4 Gamificação em desenvolvimento de software 43

III PROPOSTA 47

8 MAPEAMENTO DE MECÂNICAS DE JOGOS PARA SCRUM . . 49
8.1 Desafios e estágios de desenvolvimento 49
8.2 Métricas para *achievements* 50
8.2.1 *Achievements propostos para scrum* 51
8.2.1.1 *Achievements* de Papel 52
8.2.1.2 *Achievements* de Artefato 53
8.2.1.3 *Achievements* de cerimônias 53
8.3 Mapeamento entre desenvolvimento de software e RPG 55

9 RUPGY: UMA FERRAMENTA DE APOIO A GAMIFICAÇÃO DO SCRUM 57
9.1 Introdução 57
9.2 Funcionalidades básicas 57
9.3 Funcionalidades avançadas 58
9.4 Protótipo 59

IV AVALIAÇÃO 65

10 AVALIAÇÃO EM RETROSPECTIVA 67
10.1 *Achievements* 67
10.1.1 Resultados e discussão 68

11 SCRUM COMO RPG 73
11.1 Contextualização 73
11.2 Análise 76

12 ESTUDO DE CASO 77
12.1 Planejamento 77
12.1.1 Objetivo e questões da pesquisa 77
12.1.2 Proposições e hipóteses 78
12.1.3 Seleção de variáveis 78
12.2 Conjectura e unidade de análise 79

12.2.0.1 Métodos de coleta dados 80
12.2.0.2 Resumo da definição 80
12.2.1 Validade 81
12.2.1.1 Validade Interna 81
12.2.1.2 Validade Externa 82
12.3 Operação 83
12.4 Resultados e discussão 84

13 CONCLUSÃO 91
13.1 Limitações do trabalho 92
13.2 Trabalhos futuros 92
13.3 Publicações 93

Referências 95

V APÊNDICES 101

APÊNDICE A – ROTEIRO DE REALIZAÇÃO DA ENTREVISTA . 103

APÊNDICE B – QUESTIONÁRIO AOS DESENVOLVEDORES . . 105

Parte I

Introdução

Contextualização

O desenvolvimento de *software* pode ser entendido como um conjunto de tarefas, desde sua concepção, passando pela sua construção, até sua operação e manutenção, com base em requisitos dos usuários finais, idealmente utilizando um processo de *software* para guiar a execução dessas tarefas.

Um processo de *software* guia o desenvolvimento de um software. Esse conceito está amplamente relacionado à Engenharia de Software, sendo considerado um dos principais mecanismos para se construir um produto de alto nível de qualidade e para cumprir os contratos dentro do tempo e orçamento estimados. Vários processos e metodologias têm sido propostos ao longo das últimas décadas para melhorar a gestão da qualidade de *software*, com variados graus de sucesso. No entanto, é amplamente aceito que não existe uma bala de prata (BROOKS JR., 1987) - isto é, nenhuma abordagem resolverá todos os casos de falhas de projetos.

Em geral, os projetos de *software* que são grandes e complexos, ainda que bem especificados, envolvem aspectos desconhecidos e são vulneráveis a uma diversidade de problemas imprevisíveis. Os processos de *software* ajudam a lidar com essas situações, antecipando alternativas e expondo todas as tarefas até a conclusão do produto. No entanto, os desenvolvedores nem sempre desejam ser guiados por um processo burocrático e argumentam que essa prática pode limitar a sua criatividade. Esse fato foi um das motivações para o desenvolvimento do Manifesto para o Desenvolvimento Ágil de Software (FOWLER; HIGHSMITH, 2001).

O Manifesto Ágil motivou a criação de vários processos de *software* como o Scrum. Scrum é um modelo ágil de desenvolvimento de *software* baseado em múltiplas pequenas equipes trabalhando de forma intensiva e interdependente (SCHWABER, 2004) e é uma das abordagens ágeis mais utilizado hoje em dia (VERSIONONE, 2010).

Atualmente, a maioria das empresas de desenvolvimento de software trabalham com abordagens ágeis. Normalmente existem equipes alocadas para certos projetos que trabalham dia-a-dia na coleta de requisitos, especificação de atividades com base nos requisitos, implementação de novas funcionalidades e implantação em um produto. Essas tarefas são planejadas durante um mês e as entregas são realizadas à medida que são concluídas.

Embora processos de software possuam prescrições para atividades de desenvolvimento como elucidação de requisitos, modelagem, construção e manutenção, a má adoção dessas prescrições implica em uma série de problemas relacionados a essas atividades. Tarefas mal especificadas, falta de clareza dos objetivos de atividades e suas relações com o projeto como um todo, má qualidade do software produzido e fragilidade na comunicação entre desenvolvedores, fornecedores e clientes são exemplos desses problemas, fatores que

comumente se acredita estar diretamente relacionados a motivação em desenvolvimento de software. No entanto, um revisão sistemática intitulada *Motivation in Software Engineering: A systematic literature review* indica que ainda nã há clara compreensão do fatores que motivam desenvolvedores de software, como eles são motivados, e os benefícios e resultados da motivação na engenharia de software (BEECHAM et al., 2008).

Motivação

Muitos desenvolvedores, mesmo utilizando metodologias ágeis, têm dificuldades de seguir suas prescrições. O problema relacionado com essa questão é outro: o desenvolvimento de *software* é uma atividade desafiadora, que raramente é considerada como diversão.

Assim como muitos outros tipos de atividades, o desenvolvimento de *software* pode ser organizado como um conjunto de desafios hierárquicos e parcialmente ordenados que devem ser superados, muitas vezes necessitando de várias habilidades, diferentes conhecimentos e muito esforço de trabalho em equipe. Surpreendentemente, esse conceito é muito semelhante a uma definição abstrata de jogos: atividades em que um jogador tem de aprender novas habilidades e conhecimentos, usá-los e combiná-los para superar desafios, obtendo recompensas ou punições, dependendo do sucesso ou fracasso, respectivamente.

Muitos trabalhos têm proposto simulações e jogos sérios como ferramentas de aprendizagem ou formação em engenharia de *software* (ALVAREZ; DJAOUTI, 2011; HAUGE et al., 2011; NAVARRO, 2006; BAKER; NAVARRO; HOEK, 2005; CLAYPOOL; CLAYPOOL, 2005; SWEEDYK; KELLER, 2005). Essas aplicações lúdicas assemelham-se a um jogo por conta do uso de ambientes virtuais e personagens animados, mas muitas vezes eles não têm o fator diversão. Uma razão para isso é que eles não são construídos como jogos a partir de sua concepção e por conta disso são elaborados com regras complicadas que não consideram técnicas de *game design* já amadurecidas e consolidadas na indústria dos jogos.

Paralelo a isso, a incorporação de mecânicas de jogos em *sites* e em outras categorias de *software* que incluem interação humana tornou-se uma tendência. Essa incorporação de técnicas de *game design* em ambientes antes não considerados como jogos ou diversão é conhecida como gamificação (TAKAHASHI, 2010; CORCORAN, 2010). Acredita-se que se pode generalizar o conceito gamificação para diversas atividades dentro do desenvolvimento de *software*.

Uma vez que o desenvolvimento de software é algo desafiador, mas que em alguns momentos é considerado entediante, e ainda possui uma definição abstrata similar a de um jogo, por que não tentar transformar essa atividade em algo mais dinâmico e tentar embutir aspectos de diversão, tentando assim tornar a tarefa de desenvolver software algo mais engajante e explorar os benefícios associados a tal definição? Essa questão é o

motivador deste trabalho, que pode ser resumido em: transformar o desenvolvimento de software em uma atividade mais engajante! Esse desejo de associar aspectos lúdicos a uma atividade que corresponde a um trabalho é algo não trivial, uma vez que trabalhar é algo normalmente associado a compromissos e responsabilidades. Mas os autores deste trabalho acreditam que essa mescla pode gerar bons resultados, melhorando a vida daqueles que possuem o desenvolvimento de software como uma atividade diária.

Objetivos e contribuições

O grande objetivo deste trabalho é avaliar a viabilidade e tentar inferir os benefícios da introdução de técnicas de *game design* no contexto de desenvolvimento de software.

Como principal contribuição deste trabalho, é proposta a inclusão de conceitos de *game design* ao processo de desenvolvimento a fim de mudar a realidade associada ao desenvolvimento de *software*, transformando-o em uma tarefa mais engajante, uma vez que é possível perceber que o desenvolvimento de *software* está intimamente relacionado com um jogo, quando consideram-se as regras que regem ambas as atividades.

Como grande contribuição, apresenta-se um estudo de caso que estuda os efeitos, benefícios e desafios da introdução de técnicas de *game design* no contexto real de desenvolvimento de *software*. Outra contribuição é a uma proposta e avaliação de *achievements* individuais e de grupo utilizando dados históricos de uma equipe que trabalha em uma empresa privada.

Também são apresentados os principais requisitos de uma ferramenta de gerenciamento de projetos para incorporar as técnicas de *game design*, bem como uma ferramenta que implementa alguns desses requisitos desenvolvida para a realização do estudo de caso proposto neste trabalho.

O conhecimento da comunidade acadêmica em desenvolvimento de jogos pode contribuir para outras áreas de pesquisa aplicada e espera-se que este trabalho possa inspirar outros pesquisadores e profissionais a tentar essa abordagem em contextos diferentes.

Organização da dissertação

Esta dissertação está organizada em quatro partes divididas em treze capítulos. A Parte I (Introdução) é constituída deste capítulo, que apresenta uma introdução ao tema, a contextualização do assunto abordado, a motivação, os objetivos e as contribuições do trabalho.

A Parte II (Fundamentação Teórica) é formada pelos capítulos que apresentam a fundamentação teórica abordando o processo de software usado como base neste trabalho

como a diversão e motivação, o Scrum, o conceito de produtividade de equipes de desenvolvimento de software, além de explorar conceitos de *game design*, estudos de caso e os trabalhos relacionados.

A Parte III (Proposta) apresenta um mapeamento de mecânicas de jogos no Scrum, uma avaliação dessa abordagem em equipes reais de desenvolvimento e uma proposta de especificação de funcionalidades para uma ferramenta de gestão de tarefas que incorpore elementos de *game design*.

Por fim, a Parte IV (Avaliação) apresenta três etapas da avaliação das propostas presentes neste trabalho, que evidenciam a evolução da pesquisa. Primeiro é apresentada uma análise sobre dados históricos de uma empresa, realizada para tentar prever a aplicabilidade da proposta apresentada. Posteriormente, apresenta-se uma tentativa de aplicação de técnicas de game design durante o desenvolvimento com uma equipe real, mas ainda sem o formalismo de um estudo de caso, e por fim, é apresentado um estudo de caso desenvolvido com equipes reais, tendo como objetivo avaliar os efeitos da inclusão de elementos de *game design* no processo de desenvolvimento.

Parte II

Fundamentação teórica

Referencial teórico

Neste primeira parte do trabalho são apresentados alguns referenciais teóricos associados ao trabalho. Inicia-se com a apresentação de conceitos básicos sobre o processo de software mais praticado na atualidade e o escolhido como base para este trabalho, o Scrum. Em seguida são apresentados conceitos e métricas de produtividade de software. Posteriormente são abordados conceitos de *game design*, gamificação e diversão. Por fim é abordado a modalidade de pesquisa estudo de caso e porque ela é um formato de pesquisa aplicado ao contexto do trabalho.

1 Diversão e motivação

Introdução

O próximo capítulo apresenta teoria de um aspectos centrais que tornam os jogos influentes na vida dos seres humanos, a diversão e motivação.

1.1 Diversão

A diversão é um elemento fundamental para se estimular a realização de uma atividade. Mas afinal, o que é diversão? Por que se gasta tanta energia, tempo e dinheiro com diversão? O que a faz algo tão especial? Por que os seres humanos consideram algumas coisas divertidas e outras não?

Para responder de forma prática questões sobre comportamento humano, como explicar o que é diversão, o ponto de partida que deve-se usar é a história evolutiva humana. Para entender a evolução humana, é necessário olhar para uma data bem distante de hoje, onde houve o estabelecimento do que os humanos são hoje. Cientistas concordam que a maioria dos genes humanos foram moldados durante o tempo de seus ancestrais caçadores-coletores e os seus antecessores primatas e mamíferos. Portanto, deve-se considerar a forma como os seres humanos viviam dezenas de milhares de anos atrás para compreender o significado e o porquê da sobrevivência de muitas das suas heranças genéticas (FALSTEIN, 2004).

As diretrizes básicas para todas as criaturas são a sobrevivência e reprodução e para alguns grupos de animais (mamíferos em particular) as habilidade sociais que os permite interagir com sucesso com outros indivíduos da mesma espécie. Muito desse legado está no genes que ajudaram os ancestrais humanos a sobreviver e se reproduzir em uma sociedade de coletores e caçadores que prevaleceu desde o momento em que o homem começou a caminhar até os dias atuais (RABIN, 2005). Pessoas tendem a dividir suas vidas em tempos gastos realizando o que é necessário para sobreviver, ociosidade e períodos de prazer. Em outras palavras, trabalhar, descansar e se divertir. É fácil aceitar que o seres humanos evoluíram para atender suas necessidades básicas de sobrevivência, pois a espécie de seu potencial ancestral que não atendia a tais necessidades provavelmente foi eliminada pela seleção natural.

Considere um ancestral humano que acabou de voltar de uma caçada realizada com êxito. Ele pode descansar ou tentar caçar algo mais, trazendo assim o benefício de aumentar as reservas de recurso, mas o colocaria em risco de ser ferido ou mesmo morto

em uma caçada que não é estritamente necessária em tal momento. Outra possibilidade seria de apenas descansar, conservando suas forças não ficando disposto a realizar outra caçada até que sua 'despensa esteja vazia'. Isso o deixaria em segurança, mas traria o risco de ficar sem comida, além do fato de tornar a caçada mais difícil com o passar do tempo, por não praticar atividades que visem melhorar sua forma física (RABIN, 2005). Portanto, como teoria, acredita-se que a diversão surgiu como uma forma de se praticar uma atividade que ao mesmo tempo aliasse o prazer e o treinamento, porém com baixo risco, de modo a relaxar seus praticantes e ainda os ajudar na melhoria de condições para realização das atividades diárias. De acordo com essa teoria, a prática e aperfeiçoamento de habilidades de modo relativamente seguro, não colocando a vida em perigo, foi o grande chamariz da diversão (FALSTEIN, 2004).

Um fator também interessante para a diversão é sua explicação química. Após a realização de atividades relacionadas a entretenimento e aprendizado, os cérebros das pessoas liberam substâncias químicas conhecidas como encefalinas e endorfinas, criando efeitos e sensação de prazer e bem estar. É por conta dessa inovação evolucionária que os homens modernos procuram entretenimento e obtém prazer dele, incluindo jogos como um dos exemplos(KOSTER, 2013), uma vez que isso pode melhorar suas chances de melhor sobreviver, além de trazer um relaxamento e prazer.

1.1.1 Diversão e jogos

Algumas vezes a conexão entre um jogo específico, a forma de entretenimento e o aprendizado de habilidade para sobrevivência é difícil de ser percebida. Para um único elemento desses, pode haver muitos níveis de abstração. Por exemplo, o xadrez é baseado em movimentos de pedras em um tabuleiro, mas suas origens remetem à simulação de guerra. Já o jogo como *pac man*, combina nossas prioridade de caça, coleta e fuga!

Jogos *multiplayer* podem estender o processo de aprendizado assim que jogadores aprendem a superar estratégias dos outros e a se adaptar, o que ajuda a explicar o valor do jogo a longo prazo. Além disso, os conhecimento são ensinados em um contexto que é mais seguro do que a vida real, em particular para jogos de luta, caça e jogos de guerra, pois se eles forem tão perigosos quando a vida real, não haveria vantagem evolucionária de aprendizado de um jogo ao invés das atividades reais (KOSTER, 2013).

1.2 Motivação

A motivação humana tem sido estudada desde o início do século XX. As primeiras tentativas de entender e explicar cientificamente a motivação vieram do campo da psicometria psicoterapia, e teorias de aprendizagem (GOUVEIA et al., 2011; MORGAN; HARMON; MASLIN-COLE, 1990). É possível encontrar mais de 140 definições

formais para o termo "motivação"na literatura. No entanto, essas definições convergem para um conjunto comum de características: iniciação, direção, intensidade e persistência do comportamento (BOEHM et al., 1981).

A motivação é definida como "o processo de governança consciente para decisão entre as possíveis formas existentes de ação voluntária". É comum na literatura encontrar estudos que confundem motivação com outros conceitos como o entusiasmo, satisfação, conforto, alegria, necessidade, desejo, atitude, vontade, instinto, e fé. No entanto, a motivação é diferente desses conceitos pelas suas características humanas e sua capacidade de gerar um comportamento sustentável (BEECHAM et al., 2008).

Motivação é cada vez mais citada como um problema particularmente danoso em Engenharia de Software e apontada como uma das causas mais citadas de falhas de projetos de software (DEMARCO; LISTER, 2013).

Considerações finais

Este capítulo abordou a teoria que tenta explicar os motivos dos seres humanos gostarem de jogos. A teoria aponta a diversão como fator elementar sem o qual os jogos perderiam sua característica de engajamento e caracterizariam uma atividade enfadonha. O próximo capítulo apresenta conceitos do principal processo ágil praticado no mundo hoje, o Scrum.

2 Scrum

Introdução

Este capítulo aborda conceitos referentes ao principal processo de desenvolvimento ágil da atualidade. São apresentados a sua estrutura e suas prescrições como papéis, cerimônias e artefatos. A aplicação de técnicas de *game design* no desenvolvimento de software exige o conhecimento prévio da atividade. Por conta disso, é fundamental entender a mecânica associada ao desenvolvimento, que neste caso será baseado nas atividades prescritas pelo Scrum. Este capítulo detalha essas mecânicas e permite entender algumas das proposições realizadas com o intuito de introduzir a gamificação no processo.

2.1 Estrutura

Scrum é um *framework* ágil para a condução de projetos. Foi inicialmente formalizado para projetos de desenvolvimento de *software*, mas funciona bem para qualquer escopo de trabalho (SCHWABER, 2004).

Ao utilizar Scrum, os requisitos definidos, com a ajuda do *Product Owner*, compõem o *Product Backlog*. O *Product Backlog* é dividido em pequenas funcionalidades que podem ser entregues durante iterações de desenvolvimento de pequena duração, chamados de *Sprints*. O trabalho é passado do *Product Backlog* para um *Sprint Backlog* durante uma reunião chamada de *Sprint Planning Meeting* (SCHWABER, 2004).

A *Equipe* deve concluir o *Sprint Backlog* durante o *sprint*. Diariamente a equipe realiza as atividades Scrum, incluindo a atualização do gráfico *Burndown*. O *Sprint* termina com as reuniões *Review e Retrospective*, guiadas pelo *Scrum Master*. O resultado de um *sprint* é um subproduto que, idealmente, pode ser liberado imediatamente após a aceitação pelo *Product Owner* na reunião de revisão do *Sprint* (*Review*). Parte dessas tarefas são apresentadas na Figura 1.

Os parágrafos anteriores descrevem de forma direta e resumida o funcionamento do Scrum. No entanto, nas próximas seções iremos detalhar cada uma das três grande divisões do processo, que são os papéis, as cerimônias e os artefatos prescritos pelo Scrum. Para que uma organização utilize o Scrum como processo de desenvolvimento de software, ela deve implementar os papéis previstos pelo processo, realizar as cerimônias definidas e utilizar os artefatos que guiam o uso do processo.

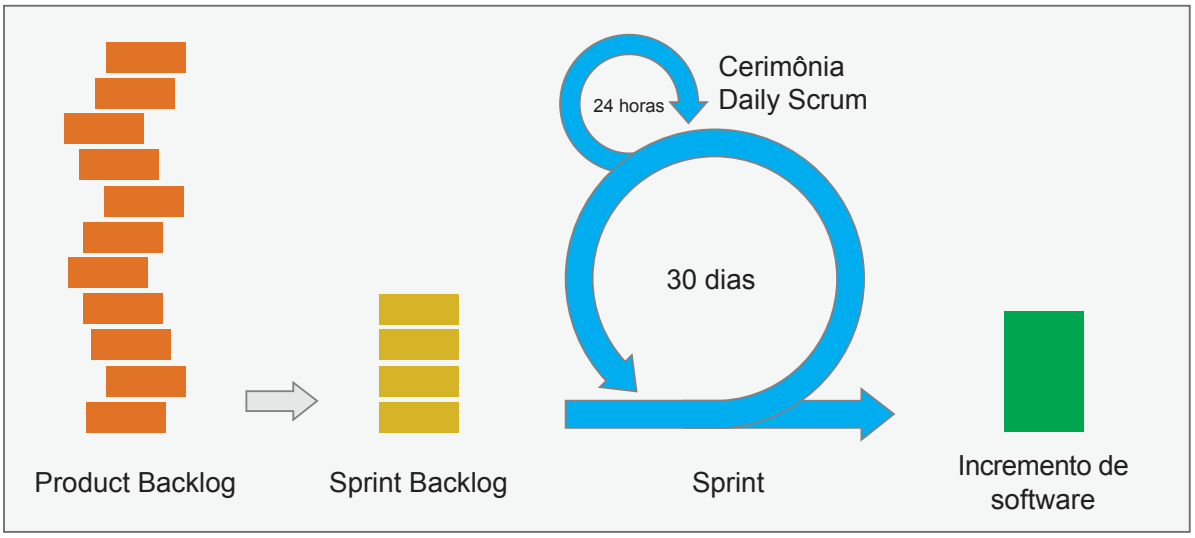


Figura 1 – Organização estrutural de partes das prescrições do Scrum *framework*.

2.2 Papéis, cerimônias e artefatos

Há apenas nove prescrições no Scrum (três papéis, três cerimônias e três artefatos) (SCHWABER, 2004). Isso significa que adotar o Scrum está associado a seguir essas nove prescrições. Por conta disso, as prescrições são a base para a gamificação relacionada à adoção do Scrum. As prescrições são:

- *Scrum Master* (SM): esse é um papel existente no Scrum. Ele representa um líder servidor, que facilita as cerimônias e supervisiona o bom funcionamento do processo Scrum. Deve garantir que as práticas do Scrum estejam sendo seguidas.

- *Product Owner* (PO): esse é outro papel existente no Scrum. Ele representa o responsável pela especificação de requisitos de alto nível, além de ser responsável por priorização das necessidades dos clientes. Seu grande objetivo é maximizar o valor entregue para o clientes e aprovar o trabalho realizado pela equipe no fim de cada *sprint*.

- Equipe (*Team*): esse é um papel associado à equipe que atua em um projeto. A equipe deve ser multifuncional, normalmente pequena (5-9 pessoas).

- *Sprint Planning Meeting* (SPM): essa é uma cerimônia realizada ao início de um ciclo de desenvolvimento, denominado Sprint. Ela é uma reunião em que o PO descreve as funcionalidades de maior prioridade para a equipe. A equipe faz perguntas suficientes de modo que seja possível transformar uma história de usuário (descrição de alto nível de um desejo), proveniente do *product backlog*, em tarefas mais detalhadas do *sprint backlog*.

- *Daily Scrum*: essa é uma cerimônia executada diariamente. Ela é uma reunião rápida cujo objetivo é apresentar a evolução dos trabalhos. Cada membro da equipe descreve o que ele realizou desde a última reunião, o trabalho que ele pretende realizar até a próxima reunião e todos os problemas ou obstáculos que possam afetar ou exigir a ajuda de outros membros da equipe.

- *Sprint Review/Retrospective* (SRR): essa é uma cerimônia realizada ao fim de um ciclo de desenvolvimento. Nela deve-se refletir sobre como se tornar mais eficaz, e em seguida, ajustar o comportamento da equipe com o consentimento de todos. Durante essa reunião a equipe avalia o *software* em execução a partir de uma inspeção (ABRAHAMSSON et al., 2002). Durante a retrospectiva a equipe faz pequenas alterações para melhorar gradativamente o produto e o processo (DERBY; LARSEN; SCHWABER, 2006).

- *Product Backlog* (PB): esse é um dos artefatos prescritos no Scrum. Ele contém itens ou requisitos de negócio, priorizados por valor de negócio. O *backlog* consiste no 'Por quê' (requisito) e no 'O Quê' (histórias de usuário). O 'Como' é determinado pela Equipe.

- *Sprint Backlog* (SB): esse é um outro artefato prescrito no Scrum, e bastante parecido com o artefato anterior. A diferença básica está no fato deste artefato conter apenas as histórias que serão desenvolvidas no sprint atual, sendo assim um subconjunto do Product Backlog. O Sprint Backlog contém histórias que não irão mudar durante o *sprint*, permitindo que a equipe possa se focar em entregar as histórias selecionadas na forma de incrementos de *software* prontos para produção.

- Gráfico *Burndown*: esse é um artefato de controle, que possibilita um gerenciamento do trabalho realizado. Ele é uma gráfico que descreve o montante de trabalho por fazer em relação ao tempo. O trabalho por fazer é apresentado no eixo vertical, com o tempo ao longo do eixo horizontal. A Figura 2 apresenta um exemplo desse gráfico. A linha diagonal azul e a vermelha representam a evolução planejada e real do trabalho ser realizado durante o *sprint*, respectivamente. Quando a linha vermelha está acima da diagonal azul, pode-se concluir que o andamento do trabalho está atrasado em relação ao que foi planejado enquanto que quando a série vermelha está abaixo da diagonal azul, isto significa que a equipe está adiantada em relação ao que foi planejado.

Considerações finais

Neste capítulo foram apresentados a estrutura, conceitos e prescrições do processo de desenvolvimento Scrum. Eles são de fundamental importância para a apresentação

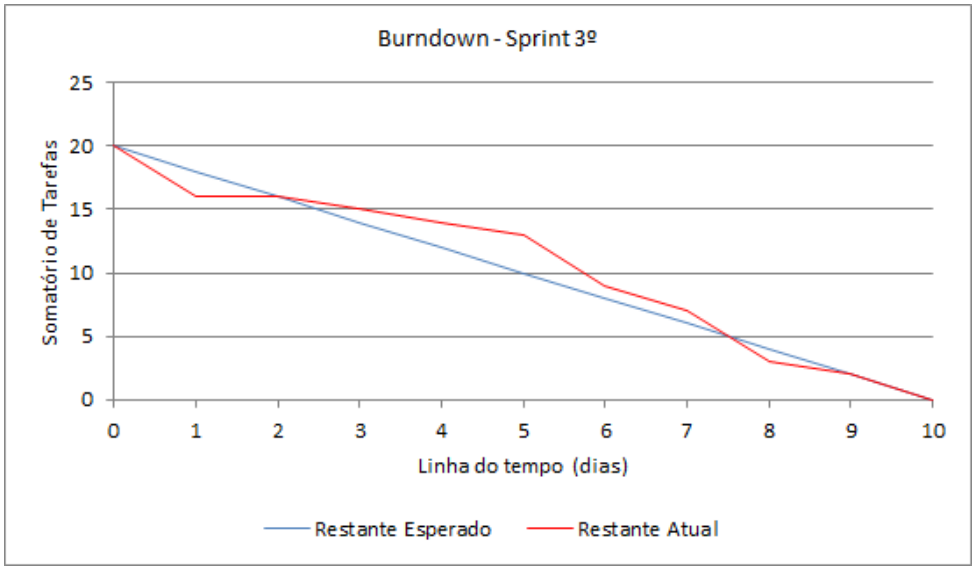


Figura 2 – Exemplo do gráfico *burndown*. Os eixos vertical e horizontal representam o montante de trabalho a ser realizado durante o *sprint* (*sprint backog*) e seu tempo de duração, respectivamente.

e compreensão da proposta de gamificação do processo de desenvolvimento de software apresentada no Capítulo 8. O próximo capítulo trata de métrica recorrente em pesquisas da área de engenharia de software, a produtividade de equipes de desenvolvimento. Serão abordados conceitos e alguns métodos de medida.

3 Produtividade

Introdução

Produtividade é um conceito recorrente e controverso na indústria de desenvolvimento de software. Este capítulo define um conceito de produtividade que será utilizado na apresentação e avaliação de um estudo de caso apresentado no Capítulo 12. Neste capítulo são apresentados conceitos e métodos de mensuração de produtividade identificando vantagens e desvantagens de cada método.

3.1 Definição

Neste trabalho, a produtividade é definida como um atributo mensurável de equipes de desenvolvimento de *software* ou de seus membros, que descreve a saída total produzida durante um determinado espaço de tempo. A Equação 3.1 representa uma função linear genérica para produtividade: razão entre a produção da equipe de desenvolvimento (ou de apenas um membro da equipe) e o esforço demandado para essa produção. Essa função é o modelo mais utilizada na definição de produtividade em desenvolvimento de *software* e base sobre a qual as métricas de produtividade tem sido desenvolvidas (BLACKBURN; SCUDDER; WASSENHOVE, 1996; BOEHM, 1987).

$$f(produção, esforço) = \frac{produção}{esforço} \tag{3.1}$$

O desenvolvimento de *software* é constituído por diferentes atividades. Tais atividades incluem, por exemplo, descoberta de requisitos, sua análise, implementação, testes e documentação (ROYCE, 1970). Por conta das diferentes naturezas dessas atividades, a saída de um projeto de *software* não consiste em apenas um determinado tipo de unidade, mas vários diferentes (PAGELS et al., 2013).

3.2 Métricas

Algumas métricas de produtividade são focadas em uma atividade particular, sendo portanto, um tipo de unidade de saída específica, enquanto que outras são projetadas para ser aplicáveis a todas atividades. Como exemplo, são consideradas medidas de desenvolvimento de software: linhas de código escritas, pontos de função implementados, pontos de história entregues, quantidade de artefatos criados ou quantitativo de tarefas concluídas.

O cálculo da produtividade baseada na quantidade de linhas de código (LoC, do inglês *lines of code*) é uma das formas mais populares de medir a atividade de codificação. Nessa métrica, o total de LoC produzido serve como a saída e o tempo gasto com a atividade de codificação como a entrada ou esforço.

A principal vantagem dessa abordagem é a facilidade de aferição: a quantidade de LoC pode ser facilmente coletada da maioria dos sistemas de controle de versão e o tempo aplicado em atividade de programação pode também ser facilmente definido ainda que com estimativas ou aproximações.

A principal desvantagem dessa métrica é que por conta dela ser baseada somente no código produzido, ela não considera atividades que não resultem em código. Outra desvantagem de utilizar o código fonte como métrica é que linguagens de programação diferem na quantidade de código escrito pelo usuário para uma mesma funcionalidade. Além disso, desenvolvedores diferentes podem ter estilos diferentes, priorizando ou não a legibilidade do código de modo a influenciar na quantidade de LoC gerada (PAGELS et al., 2013).

Um boa alternativa para LoC é medir a quantidade de vezes que os desenvolvedores "tocam"as linhas de código. Dessa forma, ao invés de contar o número de linhas, conta-se quantas vezes elas foram alteradas. Essa métrica é conhecida como *Hits-of-Code*(HoC).

A métrica HoC sempre sobe ou se mantém constante. Seu valor nunca pode ser inferior ao que era anteriormente. Assim como o esforço de trabalho, o valor da métrica Ho incrementa ou mantém-se estagnada. LoC não funciona dessa forma. Uma base de código que seja submetida a um processo de refatoração provavelmente apresentará apresentar redução ou estagnação nos valores da métrica LoC quando comparada com o valore anterior a refatoração. De forma equivocada, baseando na métrica LoC pode-se concluir que os trabalhadores não trabalharam ou passaram a trabalhar menos do que antes. De forma diferente, HoC aumenta à medida que os desenvolvedores submetem código alterado ao repositório de código, logo, o valor do HoC está objetivamente correlacionado com o esforço real de desenvolvedores que trabalham com a base de código(BUGAYENKO, 2014).

Outra métrica que pode ser utilizada para o cálculo da produtividade no desenvolvimento de *software* é a *Velocity* (Velocidade). Essa métrica pode ser determinada para os projetos em que o esforço para a realização das tarefas é estimado. O esforço pode ser denotado como uma unidade de medida comum como horas, dias, ou uma unidade abstrata, como pontos de história ou alguma categorização como "pequena", "média", "grande" e "muito grande". Dessa forma, *velocity* é definida como a razão entre o esforço estimado e o tempo consumido para realização de uma ou várias tarefas (DOWNEY; SUTHERLAND, 2013). Um exemplo dessa medida é a quantidade de pontos de história estimados entregues por dia.

O principal benefício da métrica *velocity* é que, ao contrário de medição baseado em LoC ou HoC, o modelo não depende do código fonte gerado, uma vez que o ele não é usado como entrada do modelo. Logo, os problemas com linhas de código como base para a medição são evitados (PAGELS et al., 2013).

Apesar de não ter qualquer uma das principais desvantagens encontradas para a métrica de produtividade baseada em LoC, a *velocity* possui uma desvantagem por conta das tarefas necessitarem da suas estimativas em termos do esforço necessário para concluí-las. Dessa forma, a métrica é tão precisa quanto essas estimativas (PAGELS et al., 2013).

Uma outra métrica útil para cálculo de produtividade é a Velocidade Estimada Implícita (*Implicitly Estimated Velocity* - IEV). Essa métrica baseia-se no pressuposto de que mesmo sem estimativa explícita, a atividade de divisão de escopo de trabalho em tarefas implica, implicitamente, em estimar tarefas limitadas por um limiar máximo de esforço, de modo que tarefas que ultrapassem esse limite passarão a ser dividas em subtarefas. Desse modo, a métrica IEV é caracterizada pelo número de tarefas que um indivíduo ou equipe tenham realizado durante um determinado período de tempo (PAGELS et al., 2013). A principal vantagem dessa métrica é independência da vulnerabilidade das estimativas de esforço para cada tarefa planejada.

Considerações finais

Neste trabalho, a produtividade é um aspecto para a aferição do impacto da aplicação de técnicas de *game design* no processo de desenvolvimento de *software* e será de fundamental uso no estudo de caso que será apresentado no Capítulo 12.1. Neste capítulo foram apresentados a definição de produtividade assumida e métodos para sua medição. O próximo capítulo descreve aspectos da área de jogos, abordando conceitos e mecânicas presentes no *game design*.

4 Conceitos de *game design*

Introdução

Neste capítulo são apresentados conceitos provenientes da área de jogos, incluindo a definição de mecânica de jogos, uma estrutura genérica de jogos considerando desafios, regras, punições e recompensas, além de apresentar mecânicas de jogos como *achievements*, *feedback* imediato e, por fim, uma modalidade especial de jogos, o RPG.

4.1 Mecânicas

A essência de transformar uma atividade em um jogo é torná-la ao mesmo tempo desafiadora e divertida. Mecânicas de jogos bem projetadas são os aspectos que diferenciam um jogo divertido de outros tipos de atividades de lazer, como ler um bom livro, ou assistir filmes. Existem muitas definições para a mecânica de um jogo, mas para o escopo deste trabalho, será utilizada a proposta feita pelo *game designer* Daniel Cook (COOK, 2006), que por sua vez se baseia no livro *Theory of Fun*, de Raph Kosters:

"Mecânicas de jogo são sistemas/simulações baseados em regras que facilitam e incentivam o usuário a explorar e aprender as propriedades de seu espaço de possibilidade, por meio do uso de mecanismos de feedback." (KOSTER, 2013).

Muitas outras atividades simples, quando regidas por regras de jogo ou mecânicas, podem também tornar-se atividades divertidas. Correr em um campo retangular, enquanto se chuta uma bola, na presença de outras pessoas, pode se visto como algo extremamente chato, mas quando existem regras, adversários, objetivos e recompensas, essa atividade é considerada bastante agradável para uma grande população mundial: esse é o futebol!

No processo de criação de mecânicas de jogo, o *designer* de jogos deve considerar que os desafios elaborados devem ser bem equilibrados, pois o cérebro humano perde interesse em desafios que são muito fáceis ou muito difíceis de serem superados (CHEN, 2007). Nesse contexto, um outro elemento de *game design* muito importante é o *loop* desafio-punição-recompensa, que está intimamente relacionado com a definição de mecânica de jogo acima mencionada.

Os jogos estão constantemente desafiando o jogador a superar obstáculos, que ele tenta enfrentar usando suas habilidades já aprendidas ou dominadas previamente. Quando o jogador não consegue superar um desafio, ele enfrenta a punição definida pelas regras do jogo, ao passo que quando ele obtém êxito, uma recompensa (direta ou indireta) é conseguida (CHEN, 2007). Em ambos os casos, porém, é obrigatório que o sistema apresente

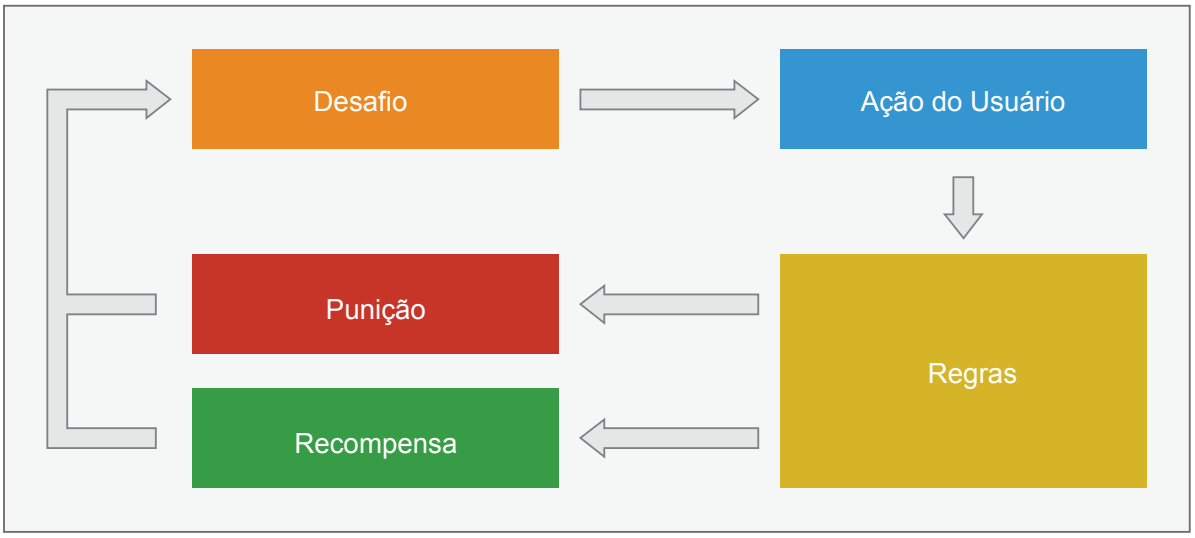


Figura 3 – O laço desafio-punição-recompensa: quando o jogador enfrenta um desafio, suas ações são confrontadas com as regras do jogo, que definem o resultado como uma recompensa ou uma punição.

um *feedback* imediato do resultado ao jogador, aumentando as chances do usuário corrigir o aprendizado até o momento (SWEETSER; WYETH, 2005). A Figura 3 ilustra como esses elementos são combinados.

4.2 Um *framework* de jogos genérico

Fundamentalmente, o prazer que se sente de jogar é baseado no fato do seres humanos serem feitos para gostar de desafios e de serem quimicamente recompensados quando utilizam habilidades (ou aprendem novos) para superar os desafios impostos pelos jogos. As seguintes premissas são aspectos necessários para que um conjunto de habilidades possa ser utilizado em um bom *game design* (adaptado de (COOK, 2008)):

- Decomposição - para facilitar o processo de aprendizagem e aquisição ou amadurecimento de uma habilidade, desafios que exigem habilidades e conhecimentos complexos devem ser decompostos em outros de menor complexidade;

- Encadeamento - o processo de decomposição anterior deve criar desafios que dependam progressivamente de habilidades e conhecimentos cada vez mais complexos, formando uma curva de aprendizagem cada vez mais desafiadora;

- Conectividade - além disso, os desafios devem depender da combinação de habilidades recém adquiridas com outras já dominadas, o que evita a criação de um conjunto disperso de desafios sem conexão.

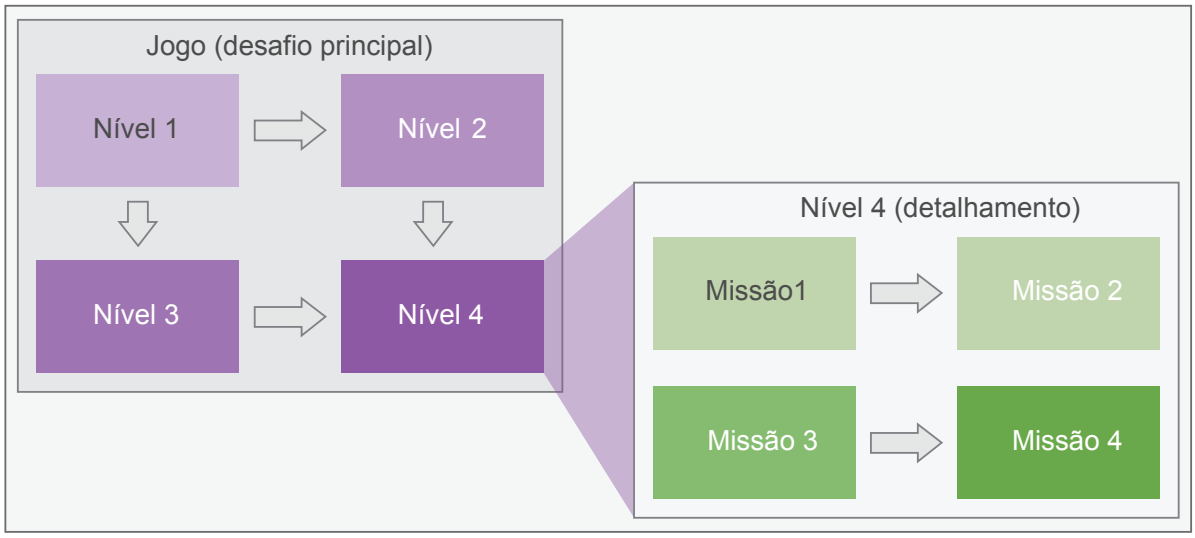


Figura 4 – Grafo hierárquico de desafios. O grande desafio (Game) pode ser encarado como uma composição de subdesafios (Level), que por sua vez podem ser subdivididos em outros subdesafios (Quest).

A estrutura para organizar um conjunto de desafios em um projeto de jogo pode ser representado por um grafo acíclico dirigido e hierárquico, no qual os nós representam desafios e as arestas indicam a relação de precedência entre os nós. Cada par nó/desafio pode ser atômico ou decomposto em um grafo mais detalhado em uma camada inferior. Por exemplo, a camada mais alta em um RPG (Seção 4.6) representa a campanha com os seus diversos níveis. Na segunda camada, os nós podem representar missões em um único nível e em camadas mais baixas, os nós são desafios atômicos que o jogador deve superar usando seu conjunto de habilidades ou combinando-as com uma recém-aprendida. A Figura 4 ilustra essa estrutura hierárquica.

O gráfico de desafio não precisa ter arestas conectando todos os nós, uma vez que a ordenação entre os subdesafios é frequentemente parcial. Muitas vezes poderia ser um conjunto de subgrafos componentes que estão desconectados um do outro. Esses conjuntos de desafios poderiam ser realizados em qualquer ordem e portanto ser superados sem ordem de precedência.

Um desafio superado pode ser definido como uma função de sucesso binária que gera 1(*verdadeiro*) para um desafio que está completo e 0 (*falso*) para os desafios que ainda estão incompletos.

Para desafios pontuais, essa função tem como entrada o estado do jogo. Sua implementação é específica para cada jogo. Para desafios compostos (de outros subdesafios), o resultado é dado também por uma expressão lógica ou matemática no conjunto de valores da função sucesso de todos os subdesafios componentes. A Equação 4.1 e a Equação 4.2 formalizam o domínio para essas duas funções, respectivamente. S indica o espaço das variáveis de estado de jogo, enquanto que F é o valor da função sucesso para cada

componente subdesafio. O termo n representa o conjunto de subdesafios. Para ambas as equações, um resultado 1 significa que desafio foi superado com sucesso, enquanto 0 significa o oposto.

$$f : S \rightarrow (1, 0) \tag{4.1}$$

$$f : F^n \rightarrow (1, 0) \tag{4.2}$$

As regras do jogo definem as expressões e fórmulas para essas funções em um jogo e são independentes do tipo de árbitro usado para calculá-las: árbitros humanos, especialistas do jogo, um *software* interativo ou mesmo os próprios jogadores.

Portanto, definir um jogo significa definir esses dois aspectos: a mecânica e as regras. Regras definem como os desafios e ações relacionam-se de modo a punir ou recompensar o jogador; a mecânica organiza essas regras e desafios em um contexto agradável (SCHELL, 2008). Às vezes, porém, esses dois aspectos não são suficientes e é preciso incrementar os jogos com outros mecanismos acessórios para engajar melhor o jogador.

4.3 *Achievements*

Achievement é um mecanismo de pontuação secundário que premia o jogador por meio do acompanhamento de como ele está atuando em uma variedade de habilidades. Podem, por exemplo, observar o uso de habilidades para resolver o mesmo tipo de desafio (normalmente um baixo nível) (HAMARI; ERANTI, 2011). São comumente concebidos com base em duas abordagens: repetição e taxa.

Um *achievement* de repetição é definido como o número de vezes que o jogador utiliza uma certa habilidade para resolver o mesmo tipo de desafio, como eliminar um certo tipo de inimigo. A Equação 4.3 mostra a fórmula para cálculo desse tipo de pontuação, na qual n denota o número total de vezes que o desafio c foi tentado e $f(c)$ é o valor para a sua função sucesso. *Achievements* de taxa são limitados a um escopo, a um desafio ou a uma porção de tempo, e, normalmente, medem a taxa de sucesso contra as tentativas totais para um dado desafio. A Equação 4.4 define a fórmula para o cálculo desse segundo tipo de *achievement*. Um exemplo de *achievement* de taxa é a porcentagem de chutes no gol executados por um jogador.

$$rep = \sum_{i=1}^{n} f(c) \tag{4.3}$$

$$rate = \frac{\sum_{i=1}^{n} f(c)}{n} \tag{4.4}$$

Achievements de repetição são frequentemente definidos com base em uma escala logarítmica aproximada, tornando cada nível subsequente cada vez mais difícil de ser alcançado. Para *achievements* de taxa, existem também vários níveis, normalmente seguindo uma escala linear de limites, muitas vezes recompensando o jogador com uma insígnia ou medalha de bronze, prata ou ouro, dependendo do limite atingido. Eles também são acumulativos, o que significa que o jogador pode ganhar muitas medalhas para os diversos *achievements*.

4.4 *Feedback* imediato

Um aspecto fundamental dos jogos (especialmente eletrônicos) é o uso de *feedback* imediato para manter o jogador consciente de seu progresso na superação ou falha dos desafios. O feedback deve retornar informações aos jogadores sempre, informando-os de onde eles estão, além de exibir uma noção de progresso, visando com isso estimular a complementação da etapa em que estão inseridos. É desejável que esse *feedback* seja dado em tempo real, o que caracteriza-o como "imediato", aumentando a sensação de imersão e também a percepção de que o jogador é o único responsável pelo resultado (SWEETSER; WYETH, 2005).

Idealmente, qualquer atividade sujeita ao uso de ideias de *game design* deve considerar a incorporação de *feedback* imediato como algo obrigatório (ZICHERMANN; CUNNINGHAM, 2011). Dessa forma, toda ação que o jogador realiza deve dar a ele uma reposta clara e nenhuma ação deve ser deixada sem "reposta". A "resposta" pode assumir várias formas. Pode ser um *feedback* visual, sonoro, ou mesmo tátil, se a plataforma, contexto ou ambiente do jogo os permitirem. Pode ser um *feedback* positivo, ou negativo, mas deve haver um *feedback*.

Feedbacks são partes essenciais de todos os jogos, e eles são evidentes na apresentação de pontuação e níveis de pontuação. Com o incremento de pontuação durante o jogo, o feedback fornece a orientação clara para o jogador de que ele está progredindo na direção "certa". Níveis e outras mecânicas de progresso ajudam a reforçar essa orientação ao reduzirem toda a progressão total de pontuação em unidades menores de mais fácil superação (ZICHERMANN; CUNNINGHAM, 2011).

Em muitos casos, tais como perda de peso ou até mesmo escrita de um livro, é difícil para um jogador entender onde ele está no início ou durante as interações iniciais de seu desafio. Além disso, a extensão e complexidade de todo o desafio é tal que, por vezes, os jogadores podem ser desmotivados pela aparente falta de progresso. Especialmente na área da saúde, educação e outros contextos onde o "desafio em questão"pode ser uma tarefa de dias, meses ou anos, o *feedback* constitui a mais importante mecânica de jogo para acompanhamento e marcação de progresso (ZICHERMANN; CUNNINGHAM, 2011).

Este aspecto do *feedback* é fundamental para a compreensão do poder e do sucesso da gamificação que será abordada no Capítulo 5.

4.5 Trapaça

O conceito ou ato de trapaça tem sido estudado de forma significativa em estudos de pesquisa acadêmica na área de jogos. Nessa área, a trapaça é definida como a violação dos limites formais e definíveis de um jogo ou uma forma injusta de obtenção de vantagens (COSTA, 2012b)

Não há uma única razão para que as pessoas pratiquem trapaças nos jogos. Existem várias razões para tal prática e que não são mutuamente exclusivas. Essas razões podem variar entre jogadores, situações, dias e jogos diferentes. Isso se dá ao fato porque, apesar de sua definição, a trapaça não é apenas sobre subverter o (jogo) sistema, mas também sobre a extensão do jogo. É uma maneira dos jogadores manterem-se jogando através de tédio, dificuldade, cenários limitados ou simplesmente jogos ruins (CONSALVO, 2005).

O conhecimento de como, quando e porquê as pessoas trapaceiam (ou se recusam a tal) é essencial para a compreensão e evolução da experiência de jogo, mas que não se apresentam como uma resposta única. As respostas variam de acordo com o contexto do jogo a depender do jogador, do jogo, da evolução atual dos jogador, dos desafios atuais, por exemplo (CONSALVO, 2005; BELL; WHALEY, 1991).

Um possível e provável problema proveniente da prática de trapaça em jogos é o descontrole do balanceamento previamente planejados entre as dificuldades dos desafios atuais e as habilidade e conhecimentos também atuais do jogador. Nesta pesquisa, a trapaça é entendida como um fator importante para seu sucesso e considerada durante as avaliações apresentadas nos capítulos 10, 11 e 12.

4.6 RPG - *Role Playing Games*

Neste trabalho propõe-se o mapeamento das prescrições do Scrum em um jogo, de forma a criar a impressão para os desenvolvedores de *software* de uma equipe que eles estão participando de um jogo da vida real. O gênero de jogo que melhor se adequa a essa situação é o *Role Playing Game* (RPG).

Um RPG é um jogo em que os jogadores assumem o papel de um 'personagem', participando de uma história de ficção narrada de forma colaborativa (TYCHSEN, 2006). A narrativa da aventura é guiada por um narrador, comumente referido como o *Game Master* (GM). O GM determina o contexto e geralmente decide o conjunto de regras a ser utilizado, além de atuar como um árbitro (TYCHSEN et al., 2005), enquanto os outros

participantes desempenham os papéis dos personagens na aventura (TYCHSEN, 2006; COPIER, 2005).

Existem alguns formatos diferentes para RPG, mas o princípio fundamental entre eles é a participação dos personagens em uma narrativa colaborativa (LINDLEY, 2005). A forma original, às vezes chamado de "RPG de papel e caneta", é realizada de forma verbal. Em um primeiro momento o GM deve descrever a situação e dizer aos jogadores o que seus personagens veem e ouvem. Depois disso, os jogadores devem descrever o que eles estão fazendo para enfrentar o desafio. O GM deve então descrever os resultados das ações de jogadores com base nas regras e definições acordadas do jogo (JACKSON et al., 2004; HENRY, 2003). De forma diferente, em um *Live Action RPG*(LARP), os jogadores executam fisicamente (de forma teatral) as ações de seus personagens (TYCHSEN et al., 2006; HARVIAINEN, 2007), ao invés de conduzir o jogo através da fala.

Diversas variedades de RPG também existem em mídia eletrônica, incluindo versões multi-usuário baseados em texto e seus sucessores, denominados *Massively Multiplayer Online Role-Playing Games* (MMORPGs) (HENRY, 2003), que são baseados em interfaces gráfica.

Considerações finais

Este capítulo apresentou conceitos da área de jogos, mais especificamente do *game design*. Nos próximos capítulos serão detalhados como esses mecanismos de *game design* e suas regras podem ser aplicadas a outras atividades do mundo real que não são comumente consideradas divertidas.

5 Gamificação

Introdução

Este capítulo aborda o conceito que permite o relacionamento entre aspectos do processo desenvolvimento de software Scrum, apresentado no Capítulo 2, com conhecimento proveniente da indústria de jogos abordados no Capitulo 4. Este conceito é conhecido como gamificação. São abordados conceitos, exemplos de aplicação e a diferença entre gamificação e outro conceito similar, os jogos sérios.

5.1 Definição

Em poucas palavras, gamificação é o uso de elementos de *game design* em contextos diferente de jogos para influenciar o comportamento, aumentar a motivação e engajamento de pessoas na realização de atividades e promover mudanças de comportamento (DETERDING et al., 2011; MARCZEWSKI, 2012). Portanto, gamificação é uma estratégia poderosa que influencia e motiva grupos de pessoas. Como exemplo, a gamificação estimula e engaja as pessoas a realizar tarefas normalmente consideradas chatas ou desafiadoras, como participação em pesquisas, compras, leitura ou prática de dietas rigorosas (CHEN, 2015).

Na última década, a gamificação começou lentamente a intrigar tanto pesquisadores e praticantes e só a partir de 2010 se tornou uma *buzzword* (ZICHERMANN; CUNNINGHAM, 2011). Hoje é largamente reconhecida como uma tendência (DYER, 2015; STOCKINGER et al., 2015; NIELSEN, 2014; SIMÕES; REDONDO; VILAS, 2013; DETERDING, 2012; TAKAHASHI, 2010).

Embora o termo gamificação seja relativamente novo, a prática já está presente há muito tempo, ainda que de forma simplificada, nos esforços de muitas empresas para alcançar novos clientes e manter os já existentes e no engajamento de funcionários através da prática de prêmios e recompensas. No entanto, foi somente nos últimos anos que a gamificação tem sido amplamente utilizada em negócios e tecnologia (SCHACHT; MAEDCHE, 2015; NEELI, 2015).

5.2 Casos e exemplos de aplicação

Existem vários exemplos de uso da gamificação em atividades presentes em nossa vida. Exemplos representativos são os já estabelecidos programas de acúmulo de milhas

aéreas. Nesses programas o participante acumula milhas (pontos ou créditos) cada vez que viaja em um voo dessas empresas ou adquire produtos e serviços de lojas parceiras. Ao alcançar uma determinada quantidade de milhas, o cliente pode trocá-las por uma nova passagem aérea em trechos nacionais ou internacionais ou então por outros produtos e serviços (OLIVEIRA; SANTOS, 2014). Esses programas visam tanto o aumento do consumo dos serviços e na formação do comportamento desejado, enquanto nesse mesmo momento os usuários procuram atingir objetivos específicos, a satisfação de que iria levá-los para ganhar recompensas (ZICHERMANN; CUNNINGHAM, 2011).

Ao utilizar práticas, técnicas e conceitos provenientes do mundo da indústria de jogos, profissionais estão tentando aprimorar os aplicativos de negócios e ferramentas de comunicação. Dessa forma, a gamificação utiliza as mecânicas de jogos para aumentar o empenho, dedicação e o prazer de jogadores em um determinado ambiente. Esses mecanismos estão diretamente relacionados com o ganho de recompensas. Em muitos casos, eles oferecem recompensas para os jogadores quando eles realizam uma ação em um tempo pré-determinado, dependendo dos níveis de dificuldade

O nível de sofisticação do ambiente promovido pela gamificação baseia-se na tarefa ou meta a ser perseguida e restrições de investimento da entidade que promove a gamificação. Em sua forma mais simples, a aplicação de técnicas de jogos pode ser uma planilha de registro usada para manter, por exemplo, a pontuação de atividades que vão desde ideias apresentadas, contatos feitos ou metas alcançadas (CHEN, 2015).

Muitas aplicações de gamificação podem ser encontradas no campo da aprendizagem, educação, crescimento e desenvolvimento pessoal. Atualmente há diversas plataformas de ensino de idiomas que não apenas se incorporam mas baseiam-se fortemente na aplicação de gamificação como Duolingo[1], Lingualeo[2], Livemocha[3] e Busuu[4]. A plataforma Duolingo, por exemplo, premia o aluno com insígnias para cada lição realizadas com sucesso. Além disso, a plataforma disponibiliza gradualmente mais lições à medida que o usuário ganha pontos para desbloquear níveis mais elevados (que na verdade são os níveis de habilidade na linguagem), além de disponibilizar um quadro de comparação entre pontuações do usuário com as de outros usuários previamente relacionados (STOCKINGER et al., 2015).

Em paralelo a abordagem de gamificação, a criação de "jogos sérios"para simular e ensinar uma atividade do mundo real é a abordagem mais simples e direta para incluir mecânicas de jogo em outros contextos. Baker et. al. (BAKER; NAVARRO; HOEK, 2003; BAKER; NAVARRO; HOEK, 2005), por exemplo, projetaram um jogo de cartas que simula o processo de *software*, visando com isso tornar um curso de engenharia de *software* mais engajante.

[1] http://www.duolingo.br
[2] http://www.lingualeo.com
[3] http://www.http://livemocha.com
[4] http://www.busuu.com

Cabe aqui explicar a diferença entre gamificação e jogos sérios. Gamificação propõe a utilização de técnicas de jogos preservando a atividade original, enquanto um "jogo sério"é uma simulação para testar ou ensinar habilidades e comportamentos aos jogadores (DANELLI, 2015; RITTERFELD; CODY; VORDERER, 2009). Apesar do fato de ambos apresentarem-se como jogos, eles requerem competências diferentes para serem projetados. Jogos sérios requerem conhecimentos técnicos (para criar um simulador de vôo é necessário ter conhecimentos sobre aeronaves) e eles não tem de ser divertidos. gamificação, por sua vez, requer conhecimentos em *game design* e se apoia no elemento diversão para engajar os jogadores (DANELLI, 2015).

5.3 O outro lado da gamificação

Embora gamificação possa ser uma excelente ferramenta para engajamento de pessoas, existem algumas ressalvas que requerem atenção. Uma crítica comum a gamificação é a sua natureza manipuladora. É reconhecido que elementos da psicologia humana são utilizados e que isso pode ser mal visto quando consideradas questões éticas. No entanto, isso é parte do que faz jogos divertidos, e o que faz os seres humanos serem atraídos por jogos. Se não fosse a característica “viciante” do jogo e a forma como os jogos permitem aos seres humanos ser descaradamente competitivos, eles não os apreciariam tanto (RUGGIERO, 2013).

Há algumas considerações quando realizada a aplicação da gamificação. Efeitos danosos, que podem ser gerados como sequência de sua má aplicação, além dos efeitos colaterais da gamificação em si, são fatores importantes que devem ser consideradas quando se utilizar gamificação como solução (COSTA, 2012a). Alguns problemas são:

- **A motivação possa ser superficial**: motivações envolvidas na gamificação podem não ser suficientes para realmente atrair usuários. Quando os incentivos não valem o esforço, não só poucos usuários irão aderir, mas os que participam irão provavelmente sentir falta o que o contexto “gamificado” realmente é em vez de simplesmente querer ser primeiro em alguma coisa.

- **Gamificação não é verdadeiramente um jogo**: Gamificação não tem a essência de um jogo. Jogos tem como caraterística criar um círculo mágico em que nada é verdadeiramente real: derrotas, mortes, vitórias e conquistas são elementos que “desaparecem” instantaneamente ao final do jogo. Quando se trata de gamificação, há um contexto real com conceitos reais e persistentes com consequências reais na vida dos “jogadores” que colocam em xeque o fator diversão.

- **Gamificação é tratada como algo de fácil aplicação**: Empresas que oferecem plataformas e serviços em gamificação vendem a ideia de que é um processo fácil e

de resultados instantâneos. Para atingir seu pleno potencial, os elementos de jogos precisam ser bem projetados e amadurecidos, um processo que pode exigir uma quantidade significativa de tempo.

Considerações finais

Este capítulo apresentou um dos principais conceitos para a realização desta pesquisa, a gamificação, além de abordar alguns exemplos de sua aplicação. O próximo capítulo apresenta um tema que é fundamental para avaliação de trabalhos que visem sua aplicação em contextos reais, o estudo de caso.

6 Estudos de Caso

Introdução

A realização de pesquisas para questões do mundo real implica na escolha ou balanceamento entre os seus nível de controle e grau de realismo. Uma situação real geralmente revela-se complexa e não-determinística, fato que dificulta seu entendimento, especialmente em estudos com objetivos exploratórios onde se almeja a identificação de relações de causalidade. Por outro lado, o aumento de controle reduz o grau de realismo do estudo, algumas vezes fazendo que fatores reais e relevantes sejam colocados em segundo plano ou mesmo fora do âmbito da pesquisa.

Neste capítulo será abordado o método de pesquisa "estudo de caso" indicado para pesquisas que lidem com situações reais e com baixo nível de controle.

6.1 Definição

Por definição, um estudo de caso é uma metodologia de pesquisa que consiste em uma investigação empírica de um fenômeno contemporâneo em um contexto real, especialmente quando os limites entre fenômeno e contexto não são claramente evidentes (YIN, 2014; RUNESON et al., 2012).

Outros exemplos de metodologia de pesquisa são o experimento controlado e o *survey*. Os estudos de caso são realizados em contextos do mundo real favorecendo o alto grau de realismo e relevância da pesquisa ainda que isso custe a redução do seu nível de controle (RUNESON; HÖST, 2009). Experimentos, por sua vez, são em sua maioria realizados em ambiente laboratorial que provê um alto nível de controle e em alguns contexto baixa relevância. *Surveys* são em geral para a compressão de toda a população estudada (não só a amostra) em cenários anteriores ou posteriores ao uso da técnica ou ferramenta pivô da pesquisa (WOHLIN et al., 2012). Como exemplos de *survey* pode-se citar pesquisa de opinião pública ou de mercado.

A natureza dos dados trabalhados na pesquisa é outro fator importante para diferenciar estudos de casos de outra modalidades de pesquisa. Dados quantitativos envolvem números e classes, enquanto dados qualitativos lidam com palavras, descrições, imagens, diagramas, etc. Os dados quantitativos são analisados por meio de ferramentas estatísticas como regressão e testes de hipótese, enquanto os dados qualitativos analisados por meio de categorização e classificação. As etapas de coleta e análise de dados em estudos de caso tendem principalmente a envolver dados qualitativos, uma vez que esses geralmente

fornecem uma descrição mais rica e profunda do fenômeno estudado (RUNESON; HÖST, 2009). Isso não impede a utilização de dados quantitativos em um estudo de caso. Na verdade, as conclusões de um estudo de caso devem ser baseadas em uma cadeia de evidências, tanto qualitativas como quantitativas, coletadas de várias fontes (RUNESON et al., 2012). Os experimentos, por outro lado, são fundamentados em dados quantitativos, praticamente não existindo a exploração de dados qualitativos.

Na realização de experimentos, as amostras ou participantes são selecionados aleatoriamente e aplicados a diferentes tratamentos. Um tratamento é a condição ou não de aplicação da variável testada do experimento sobre os participantes selecionados. Em experimentos, o objetivo é manipular poucas variáveis (preferencialmente somente uma) e controlar todas as outras a níveis fixos. O efeito dessa manipulação é medido, e com base nisso, uma análise estatística pode ser realizada. Em estudos de caso a seleção de amostras e coleta de dados é planejada e realizada de forma a aumentar sua relevância no contexto em que se dá o estudo, desconsiderando aleatoriedade como fator importante (WOHLIN et al., 2012).

Os estudos de caso não geram os mesmos resultados de relação de causalidade como experimentos controlados fazem, mas eles fornecem uma compreensão mais profunda dos fenômenos em estudo por conta de suas conclusões serem baseadas em análise de dados qualitativos e quantitativos providos por uma cadeia de evidências (RUNESON et al., 2012).

Em resumo, as principais características do estudo de caso são:

- lida com as características complexas e dinâmicas de fenômenos do mundo real;
- suas conclusões são baseadas em uma clara cadeia de evidências, seja de natureza qualitativa ou quantitativa, coletadas de várias fontes de forma planejada e consistente.
- é menos formal quando comparado a experimentos controlados.
- são mais relevantes, pois permitem entender a prática do uso de algo em seu contexto real de aplicação.

6.2 Estudos de caso e a engenharia de software

Estudo de caso é uma estratégia de pesquisa comumente usada em áreas como psicologia, sociologia, ciência política, serviço social, negócios e planejamento social. Nessas áreas, estudos de caso são realizados com os objetivos não só de aquisição de conhecimento, mas também de trazer mudanças no fenômeno em estudo (por exemplo, melhorar educação ou de assistência social). A pesquisa em engenharia de software tem objetivos de alto

nível similares de entender o porquê da engenharia de software deve ser realizada e, com esse conhecimento, tentar melhorar o processo engenharia de software, os produtos e os serviços de software gerados(RUNESON et al., 2012).

A área de engenharia de software envolve desenvolvimento, operação e manutenção de software e artefatos relacionados. A pesquisa na engenharia software tem, em grande medida, o objetivo de investigar como desenvolvimento, operação e manutenção são realizados por engenheiros de software e outras partes interessadas sob diferentes condições. Questões sociais e políticas estão envolvidas e possuem relevância para o desenvolvimento de software, pois esse é realizado por indivíduos, grupos e organizações. Dessa forma, a engenharia software é uma área multidisciplinar que envolve áreas onde os estudos de caso são normalmente realizados (RUNESON et al., 2012). Isto significa que muitas questões de pesquisa em engenharia de software são adequados para estudos de caso.

A definição de estudo de caso foca no estudo de fenômenos em seu contexto real, especialmente quando os limites entre o fenômeno e o contexto não são claros. Isso é particularmente verdadeiro na engenharia de software. Experimentação em engenharia de software já demonstrou claramente que a replicação de experimentos enfrenta muitos fatores que impactam sobre o resultado de uma atividade em engenharia de software. Estudos de caso de engenharia de software analisam fenômenos em sua configurações da vida real, onde há imprevisibilidade e baixo controle sobre variações contextuais. Por conta disso, estudos de casos, sobretudo em engenharia de software, requerem, em contraste a experimentos clássicos, uma concepção flexível onde questões de pesquisas podem ser amadurecidas durante o desenvolver do estudo (RUNESON; HÖST, 2009). Dessa forma, estudos de caso se apresentam como métodos adequados à pesquisa em engenharia de software onde seja necessário explorar ambientes reais de desenvolvimento.

Considerações

Este capítulo apresentou a definição de estudo de caso e uma comparação com outras metodologia de pesquisa, como o experimento controlado e o *survey*. Além disso, apresenta a justificativa de utilização de estudos de caso como metodologia de pesquisa em áreas como a engenharia de software. Os conceitos neste capítulo ajudarão na compreensão do estudo de caso apresentado no Capítulo 12. O próximo capítulo apresenta alguns dos trabalhos relacionados a esta pesquisa, visando com isso contextualizar este trabalho e o diferenciar dos demais já realizados.

7 Trabalhos relacionados

Neste capítulo são apresentados alguns trabalhos relacionados a esta pesquisa. Atualmente existe uma variedade de pesquisas abordando a aplicação de técnicas de *game design* em contextos diversos.

7.1 Gamificação

Uma revisão de literatura intitulada "*Does Gamification Work? — A Literature Review of Empirical Studies on Gamification*" (Gamificação funciona? - Uma revisão de literatura de estudos empíricos em gamificação) coletou resultados de estudos sobre gamificação em diversas áreas como saúde, educação, consumo sustentável e comércio. Como resposta a pergunta presente em seu título, a revisão conclui que gamificação funciona, mas com algumas ressalvas.

A revisão indicou que gamificação proporciona diversos efeitos positivos, no entanto, os efeitos são grandemente dependentes da contexto em que está sendo implementada e com deferentes períodos de duração. Além disso, seu sucesso depende fortemente das pessoas "alvo" da gamificação, pois os usuários sem encaixam em padrões de comportamentais distintos: como exemplo, para uma mecânica de tabela de classificação (*leaderboard*) alguns usuários queriam estar no topo, enquanto que para outros foi o suficiente simplesmente aparecem na tabela de classificação, independentemente de sua posição (HAMARI; KOIVISTO; SARSA, 2014).

7.2 Jogos sérios e aprendizagem

Em paralelo a abordagem de gamificação, a criação de jogos sérios para simular e ensinar uma atividade do mundo real é a abordagem mais simples e direta para incluir mecânicas de jogo em outros contextos. Baker et. al. (BAKER; NAVARRO; HOEK, 2003; BAKER; NAVARRO; HOEK, 2005), por exemplo, projetaram um jogo de cartas que simula o processo de *software*, visando com isso tornar um curso de engenharia de *software* mais engajante.

Um exemplo de de jogos sérios aplicados ao aprendizado em engenharia é a proposta apresentada no estudo intitulado "SDM – An Educational Game for Software Engineering". O estudo consiste na proposta de um jogo educativo para engenharia de software que pode ser usado para ajudar alunos na compreensão de diversos conceitos ensinados em aulas teóricas (KOHWALTER; CLUA; MURTA, 2011).

Figura 5 – Insígnias de um usuário da ferramenta *RedCritter Tracker* ganhados como recompensa por realização de tarefas.

A abordagem proposta no nosso trabalho é diferente, já que não estamos tentando simular ou ensinar uma atividade, mas sim incorporar elementos de *game design* e o fator de diversão em atividades do mundo real, nesse caso particular, no desenvolvimento de *software* orientado pelo Scrum.

7.3 Gamificação em ferramentas

Um exemplo de gamificação no contexto de desenvolvimento de *software* é o sistema *RedCritter Tracker*[1]. Ele é uma ferramenta de gerenciamento de tarefas de desenvolvimento de *software* que aplica alguns mecanismos, como recompensas e insígnias de habilidade, após realizações de tarefas.

A Figura 5 mostra um exemplo de insígnias atribuídas a um usuário do sistema. Um outro exemplo de gamificação pode ser visto no *Visual Studio Achievements*, um complemento do *Visual Studio*[2], que permite que os desenvolvedores conquistem insígnias e disputem por uma posição em um *ranking* dos melhores desenvolvedores, considerando o código que eles escreveram, o seu nível de sofisticação e os recursos do *Visual Studio* que eles utilizaram. Esse é outro exemplo de gamificação aplicada ao desenvolvimento de *software* que difere da proposta deste trabalho, que vai além da gamificação de uma ferramenta, tendo o objetivo de transformar o processo de desenvolvimento de *software* em um jogo.

[1] http://www.redcrittertracker.com
[2] http://www.microsoft.com/visualstudio

7.4 Gamificação em desenvolvimento de software

De um modo geral, os desenvolvedores gostam da ideia de serem estimulados via mecanismos baseados em gamificação. Uma pesquisa intitulada *Evaluating Developer Responses to Gamification of Software Development Practices* inclui a realização de um survey envolvendo aproximadamente 130 desenvolvedores de seis países deferentes incluindo Suécia, Estados Unidos, Suíça, Finlândia, Polônia e Índia. O survey realizado indicou que a grande maioria dos participantes (mais de 90%) são dispostos a compartilhar dados e recompensas por utilizar certas ferramentas e boas práticas de programação de desenvolvimento. Além disso, 73% do participantes afirmaram estar de alguma forma interessados na introdução de elementos de jogos em seus ambientes de trabalho(SNIPES, 2013).

Zorro (JOHNSON; KOU, 2007) é um exemplo de sistema que faz uso de gamificação para motivar a adoção de práticas por desenvolvedores. O sistema foi projetado e desenvolvido para aferir se um desenvolvedor está cumprindo as práticas de Desenvolvimento Orientado a Testes (TDD, do inglês *Test Driven Development*).

Um outro trabalho interessante incentivou alguns estudantes de ciência da computação a fazer submissões mais frequentes para o controle de versão do código fonte, ao invés de ficarem trabalhando no código em seus contextos locais por um longo período de tempo. Essa prática de sempre enviar o código para o repositório é bastante incentivada em equipes, uma vez que mantém o código atualizado e salvo em local seguro. Para incentivar a prática, foi construída uma ferramenta *web* que criava e compartilhava o *ranking* entre os desenvolvedores, baseado nas submissões realizadas (SINGER; SCHNEIDER, 2012). Esse é um exemplo de gamificação aplicada diretamente em uma prática do cotidiano de desenvolvimento e não apenas gamificação em uma ferramenta utilizada.

Em um trabalho intitulado *Understanding gamification mechanisms for software development* (DUBOIS; TAMBURRELLI, 2013), foi delineada a ideia de adotar técnicas de *game design* para engajar, treinar, monitorar e motivar todos os atores envolvidos no desenvolvimento de artefatos de *software* complexos, desde a criação, até sua implantação e manutenção. A ideia foi propor uma metodologia que pudesse ser aplicada às diferentes fases do processo de desenvolvimento. Como experimento, foi utilizado gamificação para melhorar as fases iniciais do projeto, durante o levantamento e detalhamento de requisitos de *software* em duas equipes diferentes (DUBOIS; TAMBURRELLI, 2013). O estudo permitiu concluir que embora a aplicação de gamificação no processo de desenvolvimento de *software* ter se apresentado com uma tarefa relativamente simples, prever seus efeitos se mostrou uma tarefa muito difícil de ser realizada.

Um mapeamento sistemático intitulado *"Gamification in software engineering - A systematic mapping"* foi realizado a fim de caracterizar o estado da arte quanto a

aplicação de gamificação em engenharia de software. O mapeamento foi composto de questões de pesquisa que abordaram: quais os processo da engenharia de software objeto da gamificação; quais mecânicas e elementos de gamificação propostos ou aplicados nos estudos; quais os métodos de pesquisa utilizados na condução dos estudos; e quais os tipos de tipo de pesquisa e tipo de evento o estudo foi publicado (PEDREIRA et al., 2015).

O estudo destacou que a pesquisa sobre gamificação em engenharia de software ainda encontra-se em estágio muito preliminar. Essa constatação se baseou no baixo volume de trabalhos selecionados e utilizados na pesquisa, apenas 29, e do baixo percentual de estudos publicados como artigos de revistas (*journal*). Além disso, o estudo afirma haver um baixo número de estudos com evidências claras do impacto da gamificação do desenvolvimento de software (PEDREIRA et al., 2015).

De acordo com o estudo, a maioria dos estudos focam em atividades de implementação de software, seguido de atividades da engenharia de requisitos, gestão de projetos e gerência de configuração evidenciando oportunidades de pesquisa de aplicação de gamificação em outras atividades do desenvolvimento de software. Outro aspectos destacados no mapeamento são as mecânicas utilizadas nos estudos. Os resultados apresentados apontam que mais de 38% dos estudos utilizaram apenas elementos básicos da gamificação como sistemas simplórios de pontuação (PEDREIRA et al., 2015).

Os autores do mapeamento afirmam que nos estudos que apresentam propostas de gamificação que provêm uma ferramenta "gamificada", a gamificação não foi corretamente integrada com o ecossistema, cultura e ferramentas previamente adotadas pelos desenvolvedores, equipes e organizações participantes de cadas estudo. Ainda segundo os autores, a gamificação deve ser integrada às ferramentas existentes da empresa, em vez de impor a adoção de novas ferramentas para a aplicação da gamificação. Isso fez com que a maioria dos estudos de casos enfrentassem problemas na avaliação de seus resultados e apresentem-se como estudos de difícil replicação (PEDREIRA et al., 2015) .

O mapeamento foi realizado como parte de um projeto maior (*GOAL - Gamification em Application Lifecycle*)[3] com o intuito de estudo da aplicação de gamificação em engenharia de software, em especial a gerência do ciclo de produtos de software (PEDREIRA et al., 2015).

O portal "Agile Gamification"[4] apresenta propostas de gamificação para as prescrições do Scrum. As propostas incluem pontuações e recompensa em forma de insígnias aos participantes (*product owner*, *scrum mater* e equipe) para a aspectos de atividades como participação e assiduidade e formato das cerimônias Scrum, conclusão de na realização de *sprint backlog*. Apensar de comtemplar os diversos conceitos do Scrum, a

[3] http://www.indracompany.com/it/sostenibilidad-e-innovacion/proyectos-innovacion/goal-gamificacion-orientada-a-application-lifecycle

[4] www.agilegamification.org/gamification-of-scrum/

proposta não apresenta qualquer procedimento de avaliação da aplicabilidade e sucesso da proposta (SILVA, 2015).

A abordagem proposta neste trabalho não tenta simular ou ensinar uma atividade, mas sim incorporar elementos de *game design* e o fator de diversão em atividades relacionadas ao desenvolvimento de software. Além disso, o fato de termos realizados diferentes aplicações e realizado avaliações da introdução dessas técnicas em ambientes distintos, também torna a pesquisa diferenciada às demais.

Considerações finais

Este capítulo apresentou alguns trabalhos relacionados. Embora existam trabalhos nessa mesma direção, existem muitas diferenças àquilo que foi realizado neste trabalho.

Os próximos capítulos propõem a introdução de gamificação em aspectos pontuais do processo de desenvolvimento de *software*. Isso foi a base para o trabalho e é mais explorado nas avaliações realizadas, também descrita nas seções posteriores.

Parte III

Proposta

8 Mapeamento de Mecânicas de Jogos para Scrum

Introdução

Este capítulo apresenta um mapeamento entre os conceitos do processo de desenvolvimento de software Scrum e conceitos de *game design*. É também apresentado um conjunto de *achievements* orientados a elementos do referido processo e que constituem a base para gamificação do desenvolvimento de software proposto neste trabalho.

8.1 Desafios e estágios de desenvolvimento

A fim de criar elementos de jogos (mecânicas/regras) para o Scrum, o primeiro conceito que é preciso mapear é o grafo de desafios, uma vez que esse grafo está associado ao núcleo da operação de um jogo. Sem esse mapeamento, seria muito difícil realizar qualquer introdução de gamificação no processo.

Assim, é preciso adaptar os objetivos fundamentais do Scrum como desafios para compor os nós do grafo. A Figura 6 mostra um mapeamento de Scrum para um grafo de desafios. Nesse grafo, os nós folhas representam as tarefas da equipe. Cada membro da equipe deve usar suas habilidades em requisitos, design, programação e testes para ter sucesso nesses 'desafios'. A avaliação da entrega após a conclusão para essas tarefas é normalmente feita pelo PO e os registros que indicam as conclusões devem estar registrados no gráfico *burndown*. De maneira geral, o principal objetivo é entregar uma nova porção funcional do *software*, mas isso pode ser dividido em vários objetivos menores: as tarefas do *sprint*.

A avaliação de todos os desafios de alto nível (*sprints*, liberações de novas versões do *software*), no entanto, é bem definida e pode ser formalizada por uma única função de conclusão, descrita na Equação 8.1, na qual C é o desafio a ser avaliado e c são os seus subdesafios componentes:

$$f(C) = \begin{cases} 1, & \text{se } g(c) = 1,\ \forall c \in C \\ 0, & \text{se não} \end{cases} \tag{8.1}$$

De acordo com o exposto, desafios complexos em Scrum são apenas considerados completos quando todos os subdesafios também são avaliados como completos. Às vezes, esses subdesafios têm uma ordem de precedência, como tarefas de programação dependentes

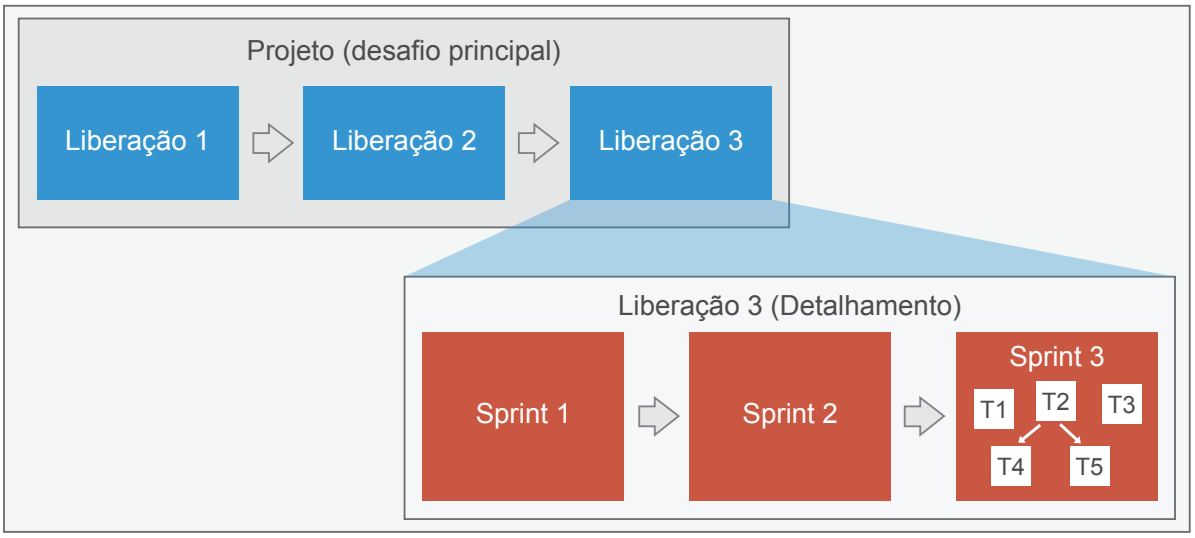


Figura 6 – Mapeamento dos componentes Scrum em um grafo hierárquico de desafios.

ou a sequência de *sprints* para uma entrega maior. Em outras circunstâncias, os desafios podem ser concluídos em qualquer ordem, tais como tarefas independentes dentro de um *sprint*, como ilustrado por *Tasks T1* e *T3* do *Sprint 3* na Figura 6.

Cada desafio presente na Figura 6, desde os de maior granularidade, tarefa, ao de maior, o projeto, impõe a necessidade de aquisição de novas habilidades e conhecimentos para que seja superado. A depender do escopo do software desenvolvido, as competências podem extrapolar as fronteiras das disciplinas genuinamente oriundas do desenvolvimento de software como linguagens de programação, interação humano-computador ou sistemas de banco de dados. Portanto, a construção do software pode exigir aquisição de competências de quaisquer áreas do conhecimento da ciência como saúde, economia e engenharia civil.

8.2 Métricas para *achievements*

Além do mapeamento direto de um projeto de desenvolvimento de *software* utilizando a hierarquia Scrum em um grafo de desafios, muitas empresas também definem e usam um vasto conjunto de métricas numéricas para medir ou mesmo premiar seus melhores desenvolvedores e equipes. Essas métricas são normalmente baseadas no desempenho de desenvolvedores em tarefas como planejamento e execução de tarefas como programação, testes ou outras atividades.

A seguir há exemplos de métricas numéricas objetivas da engenharia de *software* comumente utilizadas em gerenciamento de tarefas e análise de código fonte de *software* baseado em Scrum:

- Número de tarefas planejadas e concluídas;

- Tempo médio de conclusão de uma tarefa (estimado e real);
- Duração de cada *sprint* (planejado e real);
- Total de participações em *Daily Scrum*;
- Número de itens aprovados ou não em reunião de revisão.

Cada uma dessas métricas têm significados específicos no contexto Scrum, no entanto, simplesmente medir e expô-las não é suficiente para estimular uma equipe, por que faltam metas de referência e senso de competição. Para converter as métricas em desafio convidativo, é proposto o projeto de *achievements* individuais e de equipe. Os *achievements* individuais podem ser baseado em qualquer métrica individual, ainda que incluam tarefas de projetos diferentes. *Achievements* de equipe são limitadas aos projetos e, idealmente, devem ser baseados em métricas que contemplem cada equipe separadamente. As métricas já comentadas podem ser convertidas em *achievements*, sejam de repetição ou de taxa.

8.2.1 *Achievements propostos para scrum*

Como discutido anteriormente, os *achievements* são a representação da conclusão de algo. *Achievements* podem ser fáceis, difíceis, surpreendentes, engraçados e realizados individualmente ou em grupo. Descreve-se os *achievements* diretamente relacionados com o Scrum. A ideia principal relacionada com os *achievements* descritos é manter todos os desenvolvedores informados sobre suas realizações.

É importante ressaltar que todas os *achievements* estão relacionados a um projeto. Um projeto é um esforço temporário com começo e fim definidos. Durante um projeto, vários *sprints* podem ser realizados.

Os *achievements* propostos podem, portanto, relacionar-se com as cerimônias, artefatos e papéis do Scrum. Nas próximas seções são apresentados os *achievements* propostos. Primeiro são apresentados os *achievements* relacionados com papéis, então os relacionados aos artefatos e, finalmente, os relacionados às reuniões do Scrum.

Para cada um dos *achievements* propostos é apresentada uma premiação também proposta, como insígnias de medalha ou nível, de acordo com quão bem foram alcançados os *achievements*. A proposta de premiação consiste em uma definição inicial que deve ser evoluída e ajustada quando aplicada em um ambiente real de aplicação. É importante projetar e equilibrar os níveis (para *achievements* de repetição) e limiares (para *achievements* de taxa), a fim de torná-los interessantes. O primeiro nível e menor grau de medalha (bronze), por exemplo, deve ser fácil de ser obtido, enquanto os outros devem ser tornados cada vez mais desafiadores. Essa é a lógica para criação de um jogo, de forma a estimular os seus participantes.

8.2.1.1 *Achievements* de Papel

- **Super Scrum Master (SSM)**. O Super ScrumMaster é um *achievement* de repetição individual que representa o número de *sprints* em que todas as prescrições Scrum foram seguidos por completo. Apesar da plena implementação das prescrições não depender apenas do Scrum Master, mas de todos os envolvidos (*scrum master*, *product owner* e equipe), o *Scrum Master* é um "líder" que ajuda o resto do time Scrum a seguir o processo. Esse *achievement* tem três níveis:
  - Nível 1 (>= 1) obtido quando pelo menos um *sprint* é realizado em conformidade com todas as prescrições do Scrum;
  - Nível 2 (>= 3) três ou mais e menos de dez *sprints*;
  - Nível 3 (>= 10) dez ou mais *sprints*.
- **PO Presence (POPR)**. *PO Presence* é um *achievement* de repetição para equipes. Ele representa o número de *sprints* em que o Product Owner participou das reuniões de planejamento e revisão. Esse *achievement* possui três níveis:
  - Nível 1 (>= 1) obtido quando pelo menos um *sprint* é realizado com a participação de PO em reuniões de planejamento e revisão;
  - Nível 2 (>= 3) três ou mais e menos de dez *sprints*;
  - Nível 3 (>= 10) dez ou mais *sprints*.
- **PO Partner (POPa)**. É um *achievement* de taxa aplicado a equipes. Representa o número de *sprints* em que o PO participou de reuniões planejamento e revisão em comparação com o número total de reuniões de planejamento e revisão realizadas no projeto. Esse *achievement* tem três limiares/medalhas definidos para seus percentuais de aceitação:
  - Bronze (50%) alcançado quando 50% ou mais dos *sprints* foi realizado com a presença do PO nas reuniões de planejamento e revisão;
  - Prata (75%) 75% dos *sprints* ou mais;
  - Ouro (100%) todos os *sprints* realizados com a presença do PO.

  Esse *achievement* e todos os outros *achievements* de taxa devem ser computados periodicamente. No caso desse *achievement*, ele deve ser computado ao final de cada *sprint*. Dessa forma, assim que *sprints* sejam realizados sem a presença da PO, a taxa diminuirá fazendo com que o *achievement* antes ganho possa ser perdido.

8.2.1.2 *Achievements* de Artefato

- **Sprint Backlog Completion (SBC)**. SBC é um *achievement* de repetição aplicado a equipes e que representa o número de *sprints* nos quais todos os itens do *Sprint Backlog* foram concluídos. O *achievement* possui três níveis:
  - Nível 1 (>= 1) obtido quando pelo menos um *sprint* é concluído com a realização de todos os itens do *sprint backlog*;
  - Nível 2 (>= 3) três ou mais e menos de dez *sprints*;
  - Nível 3 (>= 10) dez ou mais *sprints*.

  Esse *achievement* deve ser sempre reavaliado imediatamente após uma reunião de revisão de *sprint* na qual o PO aceita ou desaprova a entrega de itens do *sprint backlog*.

- **Sprint Review Acceptance (SRA)**. SRA é um *achievement* de taxa aplicado a equipe que representa o número de itens aceitos pelo PO em uma reunião de avaliação em relação ao total de itens do *backlog* do *sprint* atual. Para essa *achievement* três limiares/medalhas foram definidas com o percentual mínimo de aceitação:
  - Bronze (50%)
  - Prata (75%)
  - Ouro (100%)

  Da mesma forma que o *achievement* SBC, o *achievement* SRA deve ser sempre reavaliado imediatamente após uma reunião de revisão de *sprint* em que o PO aceita ou desaprova a entrega de itens do *sprint backlog*.

8.2.1.3 *Achievements* de cerimônias

- **Clockwork Developer (CD)**. CD é um *achievement* de taxa individual que representa o número de tarefas que um desenvolvedor concluiu na duração estimada em comparação com o número total de tarefas executadas pelo desenvolvedor no *sprint*. Para esse *achievement* três limiares/medalhas foram definidos com o percentual mínimo de aceitação:
  - Bronze (50%)
  - Prata (75%)

  - Ouro (100%)

  Para uma resposta rápida e imediata, esse *achievement* deve ser avaliado sempre que o desenvolvedor concluir uma tarefa, no entanto, somente quando o desenvolvedor concluir sua última tarefa do *sprint* o *achievement* apresentará seu valor final.

- **Clockwork Team (CT)**. CT é um *achievement* de repetição para equipes e representa o número de sprints nos quais todas as tarefas foram concluídas pela equipe no tempo planejado considerando um erro de até 20% para mais ou para menos do valor estimado. Esse *achievement* tem 3 níveis:

  - Nível 1 (>= 1) obtido quando pelo menos um *sprint* é concluído com a realização de todas as tarefas de acordo suas estimativas de tempo de realização;

  - Nível 2 (>= 3) três ou mais e menos de dez *sprints*;

  - Nível 3 (>= 10) dez ou mais *sprints*.

  Os *achievements Clockwork Team* e *Clockwork Developer* são classificados como *achievements* de cerimônias porque o sucesso para realizar uma tarefa no tempo planejado está associado a um planejamento bem feito, que por sua vez deve ser realizado na reunião de planejamento de *sprint*.

- **Daily Scrum Developer Presence(DSDP)**. DSDP é um *achievement* de taxa individual que representa o número de participações de cada desenvolvedor em reuniões *daily scrum* durante um *sprint* em relação ao total de dias do *sprint*. Para esse *achievement* foram definidos três limiares/medalhas com os respectivos percentuais mínimos de aceitação:

  - Bronze (50%)

  - Prata (75%)

  - Ouro (100%)

- **Daily Scrum Team Realization (DSTR)**. DSTR é um *achievement* de taxa para equipes. Representa o número de dias que o *daily scrum* foi realizado pela equipe em comparação com o número total de dias do *sprint*. Para esse *achievement* foram também definidos três limiares/medalhas com os respectivos percentuais mínimos de aceitação:

  - Bronze (50%)

  - Prata (75%)

  - Ouro (100%)

- **Sprint Latency (SL)**. SL é um *achievement* de repetição para equipes que representa o número de *sprints* iniciados imediatamente após o fim do *sprint* anterior com tolerância máxima de 24 horas. O *achievement* possui tem 3 níveis:
  - Nível 1 (>= 1 *sprint*)
  - Nível 2 (>= 3 *sprints*);
  - Nível 3 (>= 10 *sprints*)

8.3 Mapeamento entre desenvolvimento de software e RPG

O desenvolvimento de *software* baseia-se em muitas disciplinas, cada uma com a sua própria curva de aprendizado. Quando alguém está aprendendo a programar, cada nova lição é um desafio e essa fase de aprendizagem de habilidade pode ser divertido por si só. No entanto, diferente de um jogo, um desenvolvedor experiente já tem o conjunto de habilidades necessárias para completar as tarefas atribuídas. Muitas vezes, esse fato pode tornar os desafios subjacentes um pouco chato. Nos jogos, quando isso ocorre, nós nos referimos a ele como *burnout*, o que significa que a habilidade é dominado a tal ponto que usá-la torna-se desagradável.

A fim de evitar esse efeito, projetistas de jogos normalmente envolvem o jogo a partir de histórias, definindo outras distrações para manter o jogador imerso e interessado no jogo. Uma das experiências realizadas nesta pesquisa foi o mapeamento do processo de desenvolvimento de *software* como um RPG da vida real. Como as habilidades e os desafios são reais, eles não podem ser substituídos por elementos fictícios, tal qual existe em um RPG, no entanto, alguns elementos do RPG foram mantidos, tais como:

- Pontos de Experiência - a conclusão das tarefas/desafios devem gerar pontos de experiência para os desenvolvedores que os resolvem;
- Atributos de Personagens - uma vez que as habilidades são reais e as tarefas atribuídas a um colaborador, elas normalmente estarão associadas ao seu conjunto de habilidades. Os atributos de personagens devem evoluir de acordo com o tipo de tarefas que eles concluem (níveis de atributos serão inferidos, ao invés de escolhidos);
- Classes - de maneira similar aos atributos de personagens, classes de personagens devem ser automaticamente inferidas com base nos atributos de um desenvolvedor.

Em um RPG o jogador geralmente tem a opção de tomar decisões quanto a evolução de seu personagem. Neste tipo de proposta, que é a adaptação do desenvolvimento de

software a um RPG, isso também pode ser realizado. No entanto, essa escolha acontece no mundo real e se dá quando a equipe ou desenvolvedor planejam o tipo de tarefa que cada desenvolvedor deve realizar.

A grande dificuldade do mapeamento do desenvolvimento de software para um RPG é a condução do jogo, uma vez que isso exige bastante dedicação do *game master* para manter os participantes sempre envolvidos e estimulados com os desafios impostos. Um detalhamento dessa condução é apresentada como uma das etapas de avaliação das propostas de introdução de técnicas de *game design* no desenvolvimento de software.

Considerações finais

A definição de *achievements* é um passo importante para a gamificação no desenvolvimento de software. No entanto, de nada eles servem se não forem capazes de gerar estímulo entre as equipes. Por conta disso, foi realizada uma avaliação para os *achievements* individuais e de equipes em um ambiente industrial, apenas com o intuito de avaliar se os níveis e as medalhas idealizadas seriam obtidas ao longo do desenrolar dos trabalhos das equipes. Essa avaliação preliminar é descrita no Capítulo 10. Da mesma forma, foi conduzida uma avaliação do uso do RPG como forma de estimular o desenvolvimento de software. No entanto, para que a aplicação da gamificação seja possível, é necessário que exista uma ferramenta com suporte para isso. O próximo capítulo apresenta justamente a especificação de uma ferramenta de gestão de projeto incluindo funcionalidades que contemplam as propostas apresentadas neste capítulo.

9 RUPGY: uma ferramenta de apoio a gamificação do Scrum

9.1 Introdução

O mapeamento dos *achievements* propostos ao Scrum deixou a impressão que a ideia da aplicar técnicas de *game design* ao Scrum pode ser útil para transformar o desenvolvimento de software em uma tarefa engajante. Com base nisso, foi proposto um conjunto de funcionalidades básicas que deveriam ser integradas em uma ferramenta de gestão de tarefas, visando com isso introduzir a gamificação inicial para uso em um ambiente de desenvolvimento. Essa ferramenta, denominada RUPGY, é um trocadilho relacionado ao termo RPG e *RUGBY*, uma vez que o Scrum (abreviação de *scrummage*) é uma formação usada para reiniciar um jogo de *rugby*.

O objetivo do desenvolvimento de uma ferramenta Scrum de gestão tarefas é incluir as funcionalidades de *game design* como um complemento, implementando o máximo de *feedback* possível e fazendo com que os desenvolvedores sempre estejam cientes das mecânicas em funcionamento no "jogo".

Primeiro serão listadas as funcionalidades padrão requeridas de uma ferramenta de gestão de tarefas baseada no Scrum, para que essa possa incorporar as funcionalidades de *game design* propostas. Na Seção 9.3 são sugeridos como os conceitos e funcionalidades propostas podem ser implementadas e adicionadas a esse tipo de ferramenta. A Seção 9.4 apresenta uma implementação desenvolvida orientada para a realização de um estudo de caso apresentado no Capítulo 12

9.2 Funcionalidades básicas

Nesta seção são apresentados os requisitos relacionados a uma ferramenta de gerenciamento de projeto comumente usada por equipes de desenvolvimento que adotam Scrum. O principal objetivo dessa ferramenta é registrar o planejamento e realizações de tarefas dos *sprints*. Os requisitos são apresentados como uma lista simples de itens, ilustrando os conceitos necessários, seguidos de uma explicação que detalha o requisito e seu uso.

- **Projeto, *backlog*, versão, *sprints* e tarefas**. Uma empresa de desenvolvimento de software deve sempre ter ao seu alcance todas as informações sobre seus projetos em execução. Um projeto tem vários requisitos, representados pelo *backlog*, a ser entregue

em várias iterações (sprints), cada uma composta por várias tarefas. Todas essas relações entre o projeto, o *backlog*, a versão, *sprints* e tarefas devem ser registrados, uma vez que os *achievements* propostos neste trabalho requerem a análise em seus respectivos contextos a fim de prover *feedback* imediato.

- **Registros granulares para tarefas (desenvolvedor, tempo gasto, requisitos associados, disciplina)**. É fundamental registrar o desenvolvedor associado a cada tarefa, o tempo gasto para sua realização, requisitos associados e a natureza da tarefa (disciplina, como elucidação de requisitos, codificação ou teste, por exemplo). Além disso, deve ser fácil a manipulação das tarefas como iniciar, parar, pausar e concluir uma tarefa.

- **Reuniões Scrum**. Um dos elementos chaves do Scrum são suas cerimônias. Esses eventos devem ser registrados, a fim de permitir a avaliação dos *achievements* de cerimônia propostos neste trabalho. Para cada reunião é necessário registrar quando, o quê, porquê, quem, como e por quanto tempo.

9.3 Funcionalidades avançadas

RUPGY é uma proposta para incorporar as mecânicas de RPG ao cotidiano do Scrum. O objetivo é fazer com que o desenvolvedor fique mais consciente de seu *jogador-desenvolvedor-personagem*, criando laços emocionais com esse ente virtual, melhorando assim o engajamento com suas tarefas diárias. Nesta seção são apresentadas as funcionalidades desejadas para o RUPGY.

- **Motor de características de personagens**. No RUPGY, os conhecimentos e habilidades utilizados para superar os desafios são as disciplinas da engenharia de *software* (requisitos, análise, projeto, implementação, teste, gerência de projeto) e por conta de cada tarefa estar relacionada a alguma dessas disciplinas, o objetivo é ter um sistema que calcule a experiência que o desenvolvedor obtém nessas disciplinas após concluir uma tarefa.

- **Motor de inferência de classes**. Dado que as características do jogador não são escolhidas de forma direta, as classes dos jogadores deverão ser também automaticamente inferidas baseadas nas disciplinas mais utilizadas por cada desenvolvedor (programador, testador, *scrum master*, desenvolvedor híbrido). De forma similar ao *achievements* de repetição, cada classe deverá ter escalas de limiares mínimos de pontos de experiência para alcançar diferentes *levels*.

- **Motor de *Achievements***. Na avaliação de abordagem (Capítulo 10), foram extraídos os resultados dos *achievements* diretamente da base de dados de registros de

tarefas de uma empresa privada. Para o RUPGY, esse recurso deve ser automatizado, a fim de dar um feedback imediato, sempre que um novo *level* ou medalha for obtido.

- ***Feedback* Imediato**. Desenvolvedores manipulam suas tarefas o tempo todo. Às vezes, uma manipulação pode gerar uma mudança de status na tarefa, ou mesmo no projeto. O sistema deve dar um *feedback* de forma imediata, de maneira visual e sonora, dessas alterações nos atributos do personagem.

- **Visualização de perfil de jogador**. As informações dos personagens/jogadores acima listados devem estar disponíveis para o desenvolvedor em uma tela de visualização de perfil do personagem com sua evolução histórica.

Essa é uma lista simples de funcionalidades desejadas para uma ferramenta de gerenciamento de tarefas baseado em Scrum, que inclui as mecânicas de *game design* projetadas para um processo de gamificação do Scrum. A partir do atendimento a esses requisitos seria possível pensarmos em evoluções das mecânicas utilizadas. No entanto, considera-se essa proposta bem apropriada para uma versão inicial.

9.4 Protótipo

Apresenta-se nesta seção aspectos da implementação de um protótipo em parte inspirado nas funcionalidades propostas.

O protótipo foi concebido especialmente para a realização do estudo de caso apresentado no Capítulo 12, que foi realizado em uma empresa real de desenvolvimento de software. Portanto, a definição e evolução das funcionalidades do protótipo basearam-se no contexto específico dos projetos e práticas dessa empresa. Uma limitação ao estudo deveu-se ao fato da empresa utilizar uma ferramenta para gestão de projetos e tarefas que não é completamente adequado ao Scrum, fato esse que impediu o registro de diversas das cerimônias realizadas. Esse fato também limitou o conjunto de *achievements* desenvolvidos, uma vez que só poderiam ser criados os *achievements* associados a dados presentes na ferramenta usada pela empresa. Por conta da empresa já adotar o *Redmine* [1] como sua ferramenta de gestão de projetos, o protótipo foi implementado como um componente plugável para essa plataforma.

O *Redmine* é uma ferramenta *web* de código aberto para gestão de projeto. Ele permite aos seus usuários gerenciar múltiplos projetos e subprojetos associados. Possui funcionalidades como *wiki* e fórum de projeto, controle de tempo gasto em tarefas, versão de software, iterações, *backlog*, calendários e gráficos de *gantt* para ajudar representação visual dos projetos e seus prazos, além de ser integrável a vários sistemas de controle de

[1] https://www.redmine.org

versão. Grande parte do poder do *Redmine* provém da sua potencialidade de configuração e extensão (LESYUK, 2013). O registro de projetos e tarefas permite a criação de novos campos e alteração do fluxo de trabalho (marcação de status de tarefas como iniciada, realizada, concluída...), além de permitir a inclusão de componentes reusáveis desenvolvidos por sua comunidade de usuários, ampliando sua aplicabilidade aos diversos contextos e especificidades de projeto (BEVILACQUA, 2014).

Foram implementadas funcionalidades como funções de pontuação, insígnias de tarefas premiadas, notificações, classificação e visualização de perfil de jogador.

A funcionalidade de pontuação desenvolvida consistiu em dois atributos: *xp* e *level*. *Xp* e *level* são mecânicas de pontuação utilizada para quantificar a progressão de um jogador, sendo muito popular no mundo dos jogos virtuais e presente em uma vasta quantidade de jogos. O *xp*, do inglês eXperience Points (pontos de experiência), é um contador que representa a progressão de pontos de experiência do jogador. Iniciando em zero, seu valor é incrementado no momento de registro de uma tarefa como "executada"e é decrementado quando tarefas que provocam seu incremento são canceladas. O *level* é também um contador que representa um estágio de progressão de experiência. Quando o valor do *xp* alcança patamares pré-definidos, o valor do *level* é incrementado e o valor do *xp* retorna a zero. A definição dos valores dessa relação foi arbitrariamente realizada e balanceada (redefinição da relação entre *xp* e *level*) à medida que os desenvolvedores do estudo de caso progrediram seus *xp* e *level*. A Tabela 1 apresenta um exemplo de relação entre *xp* e *level*. Quando que jogador possuir o *level* 1 terá de ganhar 20 pontos de experiência (xp) para obter o *level* 2. Após alcançar o level 2, o jogador precisará de mais 40 pontos para progredir ao *level* 3.

Tabela 1 – Exemplo de valores da relação entre Xp *Level*.

level	**xp**
1	20
2	40
3	80
4	160
5	320

O valor de incremento de *xp* para cada tarefa realizada é calculado por uma função composta definida a partir dos atributos tempo estimado e tempo gasto da tarefa. A Equação 9.1 e a Equação 9.2 apresentam essa função onde e e g são, respectivamente, o tempo estimado e gasto da tarefa, C é uma constante utilizada para o balanceamento da função e *trunk* é uma função auxiliar que calcula o porção inteira de um valor real. Para tarefas que tiveram seu custo estimado, o valor do incremento corresponde ao valor inteiro do produto entre a constante de balanceamento e o tempo estimado, enquanto que tarefas

realizados sem registro de estimativa, o valor é o produto entre a constante e metade do tempo gasto.

$$f(e,g,C) = \begin{cases} g(e,g,C,0,1/2), & \text{se } e = 0 \\ g(e,g,C,1,0), & \text{se não} \end{cases} \tag{9.1}$$

$$g(e,g,E,G) = trunk(1 + C(eE + gG)) \tag{9.2}$$

Cada *level* é associado a uma cor: level 1-verde; 2-laranja; 3-vermelho; 4-azul; 5-cinza. O *xp* e a informação de *level* aparecem permanentemente na seção superior do *redmine*, como pode ser visualizado na Figura 7.

Figura 7 – Fragmento de tela do *redmine* com o protótipo desenvolvido mostrando apresentação de *xp* (valor 13), *level* (valor 3) e insígnias obtidas na realização de tarefas.

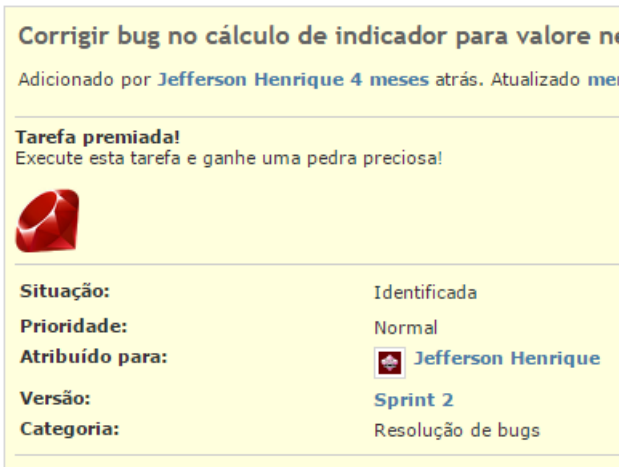


Figura 8 – Exemplo de visualização de uma tarefa premiada com uma insígnia.

A funcionalidade de insígnias de tarefas premiadas é caracterizada através da apresentação e ganho de insígnias orientadas a metáfora de pedras preciosas. Tarefas marcadas como premiáveis pelo usuário moderador, apresentam em sua "tela" de visualização, um ou mais insígnias de pedras preciosas, conforme mostrado na Figura 8. Ao executar uma tarefa premiável, o desenvolvedor ganhará mais uma insígnia, que permanecerá associada ao seu perfil no jogo e exibível permanentemente na seção superior do *Redmine*, conforme ilustrado na Figura 7, junto à informação de *xp* e *level* correntes. Os valores de *xp* e *level* atuais do desenvolvedor são 13 e 3, respectivamente, o valor 13 e o valor 80 representa o valor que o *xp* deve alcançar para a obtenção do *level* 4.

A funcionalidade de notificação cumpre o papel de *feedback imediato* quando o desenvolvedor ganha novos *xp*, altera de *level*, ou ganha insígnias de tarefas premiadas. As

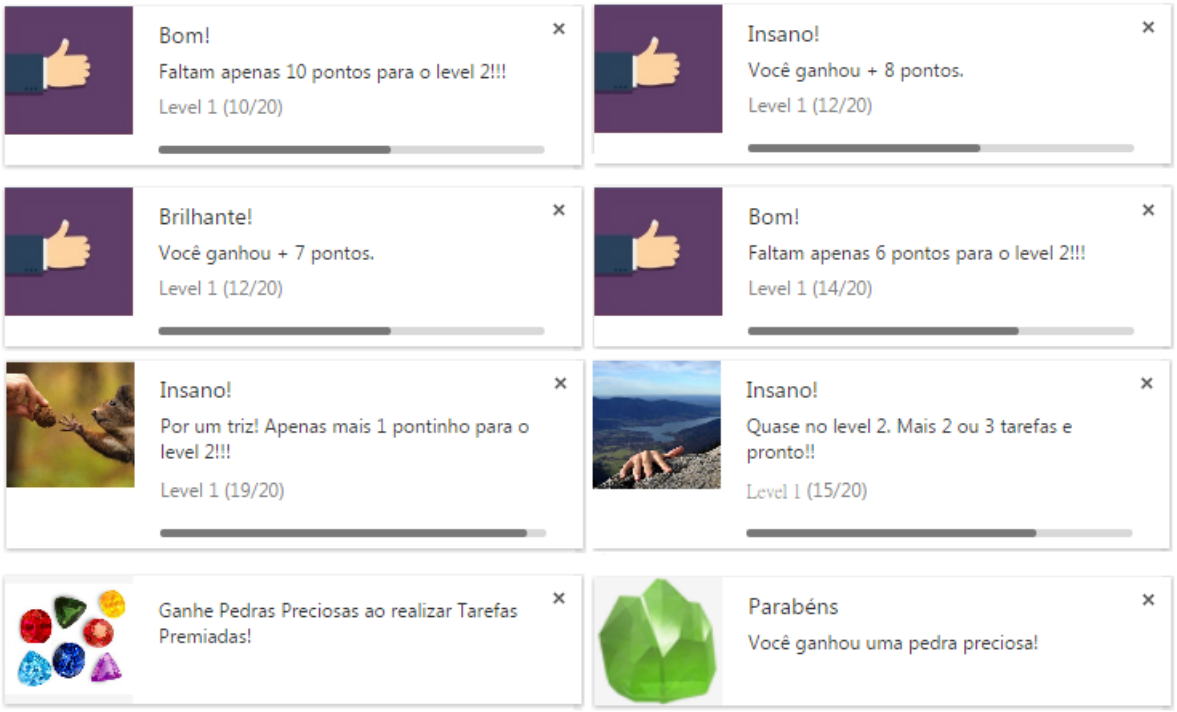


Figura 9 – Exemplos de notificações apresentadas em eventos como alteração de *xp*, *level* e obtenção de insígnias.

notificações se apresentam como um *pop up* temporário composto de elementos associados aos eventos responsáveis por gerar a notificação. A Figura 9 exibe alguns exemplos de visualização de notificações.

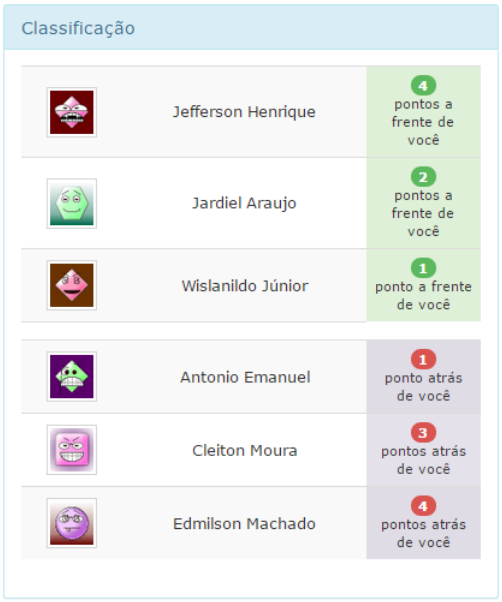


Figura 10 – Fragmento da tela ilustrando a visualização de um quadro de classificação de desenvolvedores.

A fim de permitir que um desenvolvedor saiba seu status em comparação aos demais desenvolvedores, foi criada a funcionalidade denominada *classificação*. Ela apresenta um

quadro com os desenvolvedores com pontuações(*xp* e *level*) imediatamente superiores e imediatamente inferiores. A não exibição da listagem de todos os jogadores é uma tentativa de reduzir efeitos colaterais da exposição direta de desenvolvedores em "últimas posições". A Figura 10 apresenta um exemplo de quadro de classificação.

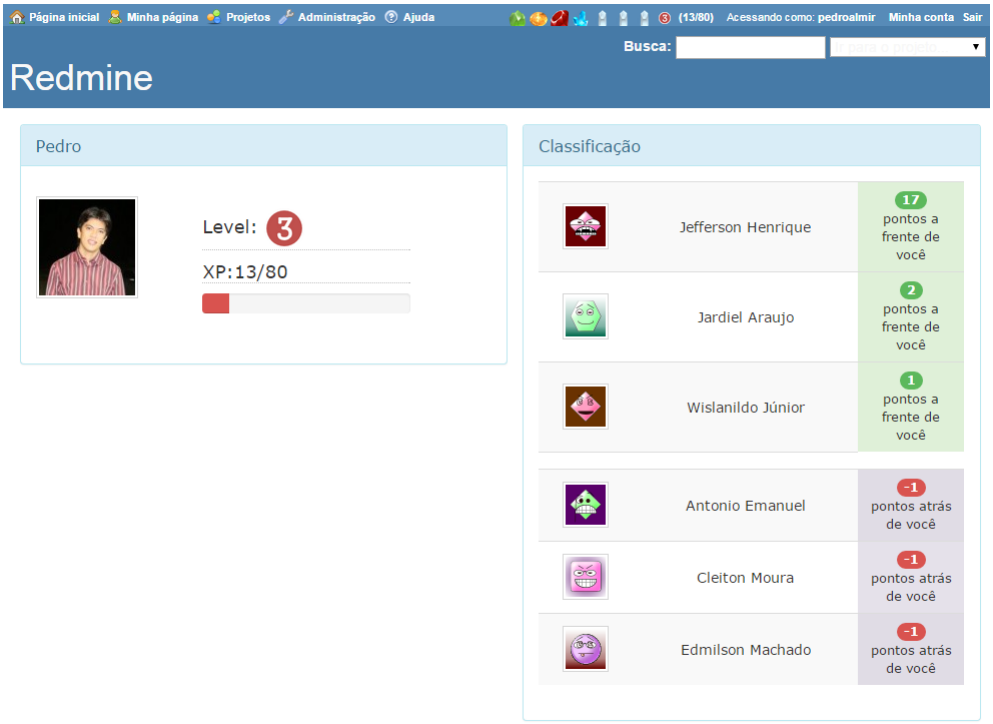


Figura 11 – Visualização de perfil de jogador.

A funcionalidade de visualização de perfil de jogador reúne as informações de pontuação e classificação em uma única tela de visualização de acesso restrito ao desenvolvedor correspondente. Há a informação de *xp* e *level* correntes, acompanhadas de uma barra de progresso, insígnias obtidas e classificação. A Figura 11 é uma exemplo de tela de visualização de perfil de jogador.

É importante destacar que o sucesso das funcionalidades propostas e implementadas dependem da atuação de um usuário moderador que as torne efetivas através de sua condução e balanceamento.

Por ter sido desenvolvido como um componente plugável ao *redmine*, o protótipo assume características de projeto que exigem o *redmine* como ferramenta de gestão de projeto adotada impossibilitando a sua aplicação em outros contextos. Em contra partida, essa dependência à plataforma *redmine* facilitou seu desenvolvimento e sua aplicação em equipes de desenvolvimento participantes do estudo de caso apresentado no Capítulo 12, visto que essas equipes já utilizavam o *redmine* como plataforma de gestão de projetos. É importante destacar que, apesar dessa limitação, uma contribuição maior desta pesquisa é a proposição de requisitos a uma ferramenta de gestão de tarefas para gamificação do processo de desenvolvimento.

Considerações finais

Neste capítulo foram apresentadas a proposta de especificação para uma ferramenta suporte a gamificação do processo de desenvolvimento de uma empresa. Foi apresentado um protótipo da ferramenta desenvolvida especificamente para a realização do estudo de caso que será apresentado no Capítulo 12. Os próximos capítulos apresentam os diversos estágios da "avaliação" das ideias até aqui apresentadas.

Parte IV

Avaliação

10 Avaliação em Retrospectiva

Introdução

Conforme comentado anteriormente, a definição de *achievements* e outras mecânicas de jogos no desenvolvimento de software somente pode ser considerado algo efetivo se isso gerar estímulo dentre os membros das equipes. Por conta disso, realizou-se avaliações das ideias aqui propostas, visando averiguar se as definições apresentadas seriam aplicáveis e se indicariam estímulos de engajamento em uma equipe. Esta foi a primeira avaliação realizada. Neste capítulo será descrita essa avaliação inicial e suas conclusões, que direcionaram avaliações adicionais.

10.1 *Achievements*

A aplicabilidade da abordagem apresentada foi avaliada a partir da avaliação de dados históricos de equipes reais de desenvolvimento que já utilizavam o Scrum como seu processo de desenvolvimento. Isso foi feito em parceria com a empresa Infoway. A Infoway [1] é uma empresa de tecnologia de médio porte. Ela possui hoje cerca de 30 colaboradores (programadores, *designers*, gerentes de projeto e técnicos em infraestrutura) atuando na área de desenvolvimento, divididos em quatro equipes que utilizam o Scrum desde 2008.

A empresa possui um sistema de gerenciamento de projetos baseado em Scrum que não possui funcionalidades que incorporem ou caracterizem elementos de *game design*. O sistema possui informações como registros sobre algumas das reuniões do Scrum, incluindo planejamento, revisão e retrospectiva, porém sem tanta profundidade nesses registros. Há também informações sobre o *sprint backlog* e tempos estimado e gasto para realização de tarefas, agrupadas por *sprints* e projetos, além da categorização das tarefas, incluindo testes, reuniões, *brainstorms* ou atividades de programação. Essas informações foram utilizadas para avaliar a proposta de no Scrum em um caso real.

Neste capítulo será apresentada a avaliação de alguns dos *achievements* propostos utilizando o banco de dados histórico da Infoway. Por conta de termos utilizado uma base de dados histórico para a avaliação da abordagem, o *feedback* imediato não pôde ser avaliado. Dessa forma, essa avaliação compreende aferição de quatro *achievements* baseados nas métricas coletadas a partir da base de dados. A ideia associada a essa avaliação era descobrir se os *achievements* criados seriam alcançados pelas equipes, em diferentes momentos e com deferentes intensidades, de forma a possibilitar um engajamento

[1] http://www.infoway-pi.com.br

entre as equipes, por conta das notificações que seriam disparadas quando os limiares fossem atingidos.

10.1.1 Resultados e discussão

A avaliação realizada considerou quatro *achievements*: um *achievement* de taxa para desenvolvedores e três *achievements* de repetição para equipes.

Como mencionado anteriormente, foram utilizados dados históricos para calcular os *achievements* e premiar quatro equipes de projeto, compreendendo um total de 16 colaboradores à época dessa avaliação, que utilizou dados referentes a um período de tempo de sete meses, que abrangeu entre nove a doze *sprints* dos quatro projetos.

A diferença na quantidade total de *sprints* para cada projeto (entre 9 e 12 *sprints*) existe por que cada equipe pode, eventualmente, trabalhar com *sprints* de duração diferente, além de poder haver intervalos de tempo diferentes entre *sprints*.

O resultado da *achievement Clockwork Developer* é apresentado na Tabela 2. Há a indicação do número de medalhas que seriam entregues para cada colaborador, de acordo com seu projeto. Os desenvolvedores poderiam ter ganho no máximo até dez medalhas de ouro no período considerado. A partir da análise, pode-se observar que somente um desenvolvedor ganhou sete medalhas e todos os outros desenvolvedores ganharam ao menos uma medalha.

Em projetos em que os desenvolvedores trabalharam para a mesma duração de tempo, grandes diferenças no número de medalhas entre os membros da equipe podem indicar problemas de alocação desequilibrada de tarefas (alguns membros da equipe sempre realizam as tarefas mais complexas, enquanto outros mais fáceis) ou deficiência técnica de alguns membros da equipe. No Projeto I, por exemplo, enquanto os desenvolvedores A e C conseguiriam muitas medalhas, os desenvolvedores B e D teriam conseguido apenas três medalhas.

A Tabela 3 apresenta os níveis registrados para cada equipe considerando o *achievement Clockwork Team*. Todas as equipes alcançaram ao menos o Nível 1, o que significa que todos as equipes concluíram ao menos um *sprint* como planejado. As equipes dos Projetos II e III concluíram dois *sprints* dentro de planejamento, de um total de nove e doze *sprints* realizados respectivamente, mas esse fato não assegurou um avanço de nível. Apenas a equipe do Projeto 1 alcançou o Nível 2, por ter concluído seis *sprints* dentro do templo planejado.

Um ponto interessante é que apesar ser constituída por desenvolvedores com grandes resultados para o *achievement Clockwork Developer*, a equipe do Projeto IV obteve um desempenho ruim para o *achievement* Clockwork *Team*. Isto pode representar uma integração ruim da equipe, resultando em *sprints* falhos em que a equipe foi incapaz de

Tabela 2 – Resultados individuais para o *achievement clockwork developer* agrupados por projeto: total de medalhas ganhas por cada desenvolvedor durante os *sprints* considerados nessa avaliação.

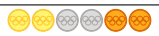
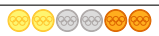
Clockwork Developer		
Projeto	**Desenvolvedor**	**Medalhas**
Projeto I	Desenvolvedor A	
Projeto I	Desenvolvedor B	
Projeto I	Desenvolvedor C	
Projeto I	Desenvolvedor D	
Projeto II	Desenvolvedor E	
Projeto II	Desenvolvedor F	
Projeto II	Desenvolvedor G	
Projeto III	Desenvolvedor H	
Projeto III	Desenvolvedor I	
Projeto III	Desenvolvedor J	
Projeto III	Desenvolvedor K	
Projeto IV	Desenvolvedor L	
Projeto IV	Desenvolvedor M	
Projeto IV	Desenvolvedor N	
Projeto IV	Desenvolvedor O	
Projeto IV	Desenvolvedor P	
Projeto IV	Desenvolvedor Q	

Tabela 3 – Resultados para o *achievement* Clockwork Team: níveis alcançados para cada projeto das equipes e o número total de *sprints* concluídos como planejado.

Clockwork Team	
Projeto	**Nível**
Projeto I	Nível 2 (6/9 sprints)
Projeto II	Nível 1 (2/9 sprints)
Projeto III	Nível 1 (2/12 sprints)
Projeto IV	Nível 1 (1/10 sprint)

entregar as funcionalidades no fim da cada *sprint*.

A Tabela 4 apresenta os níveis para o *achievement Sprint Backlog Completion*. Apenas a equipe do Projeto IV não atingiu o *Nível* 2. A equipe do Projeto II conseguiu cumprir todo o *sprint backlog* em seis *sprints* de um total nove *sprints*. Foi a melhor equipe nesse *achievement*, no entanto, ela não alcançou o *Nível* 3.

As baixas quantidade de sprints concluídos com os baixos níveis alcançados para o *achievement Sprint Backlog Completion* podem representar alguns aspectos importantes do processo de desenvolvimento adotado. O *sprint backlog* provavelmente está sendo subestimado pelos desenvolvedores, causados possivelmente pelo não esclarecimento de dúvidas referente aos requisitos com o PO, ou o time não tem um PO definido, causando

Tabela 4 – Resultados para o *achievement Sprint Backlog Completion*: níveis alcançados para cada equipe dos projetos e o total de *sprints* com o *sprint backlog* concluído.

Sprint Backlog Completion	
Projeto	**Nível**
Projeto I	Nível 2 (3/9 sprints)
Projeto II	Nível 2 (6/9 sprints)
Projeto III	Nível 2 (3/12 sprints)
Projeto IV	Nível 1 (2/10 sprint)

erros de planejamento durante os *sprints*.

A Tabela 5 mostra os níveis registrados para o *achievement Sprint Latency*. Apenas a equipe do Projeto I não alcançou o *Nível 2*. As equipes dos Projetos III e IV conseguiram iniciar cinco *sprints* sem latência entre os *sprints* de seus projetos. Apesar de apresentarem o melhor desempenho para esse critério, eles não alcançaram o *Nível 3*.

Tabela 5 – Resultados para o *achievements Sprint Latency*: níveis indicando ausência de intervalo entre *sprints*.

Sprint Latency	
Projeto	**Nível**
Projeto I	Nível 1 (1/9 sprints)
Projeto II	Nível 2 (3/9 sprints)
Projeto III	Nível 2 (5/12 sprints)
Projeto IV	Nível 2 (5/10 sprint)

Baixos resultados para esse *achievement* mostram uma dificuldade em iniciar um próximo *sprint* depois de terminar o anterior. No Projeto I, por exemplo, durante nove *sprints* apenas um *sprint* iniciou logo após o término da anterior. Isso geralmente é causado pela ausência de um PO para apoiar a definição do *Sprint Backlog*, ou mau desempenho do *Scrum Master* para assegurar que a equipe não trabalhe em tarefas durante um tempo após a término do *sprint*.

A Figura 12 mostra uma comparação entre projetos, considerando os *achievements* avaliados. A equipe do Projeto I é a melhor em dois *achievements*, mas não possui uma pontuação destacada das equipes dos outros projetos. A equipe do Projeto IV tem o segundo maior número desenvolvedores com *achievements Clockwork Developer*, no entanto, tem as piores pontuações para dois *achievements* computados para equipe.

Curiosamente, devido a essa aparente inconsistência, a empresa decidiu fazer mais investigações para entender as razões por trás desse comportamento. Isso levou à conclusão de que o uso de *achievements* não só pode ajudar a envolver as equipes a fazer o seu trabalho, mas também pode ajudar a monitorar, controlar e melhorar o desenvolvimento

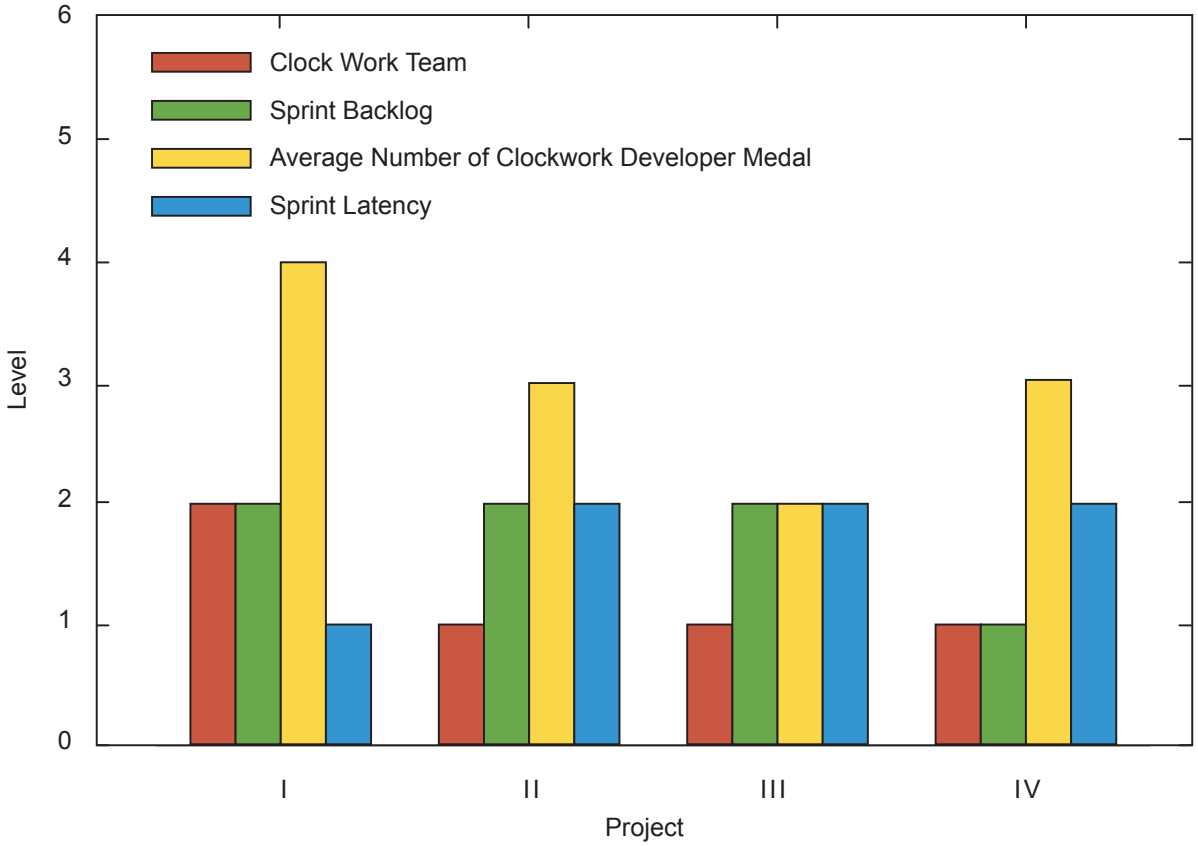


Figura 12 – Comparação entre as equipes dos projetos quanto aos resultados obtidos para os *achievements* propostos.

utilizando Scrum. Diante dos resultados obtidos, todos os membros da equipe mostraram-se interessados por novos avanços deste trabalho.

Apesar de ter dez *achievements* propostos, o banco de dados utilizado no estudo não possuía informação suficiente para a computação de todos os dez *achievements*, sendo possível contabilizar apenas quatro deles. Como exemplo desse fato, não havia informações tais como registros de participação PO em reuniões de planejamento ou registros de realização de reuniões *daily scrum* que permitissem o cálculo dos *achievements* *PO Presence* e *Daily Scrum Team Realization*, respectivamente.

Considerações finais

Este capítulo apresentou uma primeira avaliação das propostas apresentadas neste trabalho. Foram utilizados dados históricos de realização de tarefas de uma empresa de desenvolvimento de software para investigar a aplicabilidade dos *achievements* propostos no Capítulos 8. Com a realização da avaliação foi possível inferir que o uso dos *achievements* em um projeto real poderia gerar diversos *feedbacks* oriundos do alcance de medalhas e de níveis propostos nesta abordagem de . Esse resultado foi importante para a pesquisa, uma

vez que isso serviu para validar a necessidade de se fazer uma avaliação que envolvesse um projeto em execução, para a inferência de dados em tempo real, visando com isso estimular os participantes do projeto. Mas antes dessa avaliação, foi realizada uma avaliação prévia, visando entender se uma equipe se estimularia apenas com a introdução de conceitos de RPG em suas tarefas diárias de desenvolvimento. O próximo capítulo apresenta o relato dessa avaliação preliminar.

11 Scrum como RPG

Introdução

Diferente da avaliação apresentada no Capítulo 10, esta avaliação foi realizada com uma equipe de desenvolvimento real dentro do cotidiano de desenvolvimento de software da equipe. Apesar de não seguir nenhum protocolo de pesquisa, ela foi importante para os pesquisadores deste estudo avaliarem a aplicação real de técnicas de *game design* e coletaram *feedback* dos participantes, ainda que informalmente, e auxiliar na preparação do estudo de caso apresentado no Capítulo 12

11.1 Contextualização

A avaliação realizada consistiu em transformar o desenvolvimento de *software* de uma equipe com três desenvolvedores em um RPG. A avaliação teve uma duração de três semanas (duração do *sprint*) e envolveu uma equipe formada por três desenvolvedores profissionais. A equipe já trabalhava em um projeto de desenvolvimento de *software* há aproximadamente seis meses em uma empresa pública do estado do Piauí.

Essa avaliação não utilizou uma ferramenta ou aplicação que abordasse os requisitos a uma ferramenta para aplicação de do processo de desenvolvimento de software apresentados no Capítulo 9.

Como mecânicas, foi utilizado um mapa (Figura 13), uma história, em que cada personagem representou um dos desenvolvedores e um quadro com as pontuações ganhas de cada desenvolvedor a partir da conclusão de tarefas.

O mapa, apresentado na Figura 13, nada mais é do que um gráfico *burndown* (Seção 2.2) com alegorias de mapa. A diagonal "planejamento" é representada por um rio. O desejado é que a realização das tarefas ocorra como o planejado, ou ao menos não ocorra com atrasos. Por isso, a região no mapa acima do rio (acima da diagonal planejamento), que está associada a um atraso no andamento do projeto, possui elementos que, dentro do contexto normal, não são atrativos aos personagens, por serem insalubres, como desertos, montanhas e região oceânica.

As histórias foram construídas a cada dia durante o *sprint*, de modo que o desenrolar foi definido pelos acontecimentos do cotidiano de desenvolvimento da equipe, tais como a conclusão de tarefas, dificuldades enfrentadas, ausências de membros da equipe e por surgimentos e resoluções de *bugs*. No início de cada dia de trabalho, um novo capítulo da história era enviado pelo *Game Master* por *e-mail* a todos os integrantes da equipe.

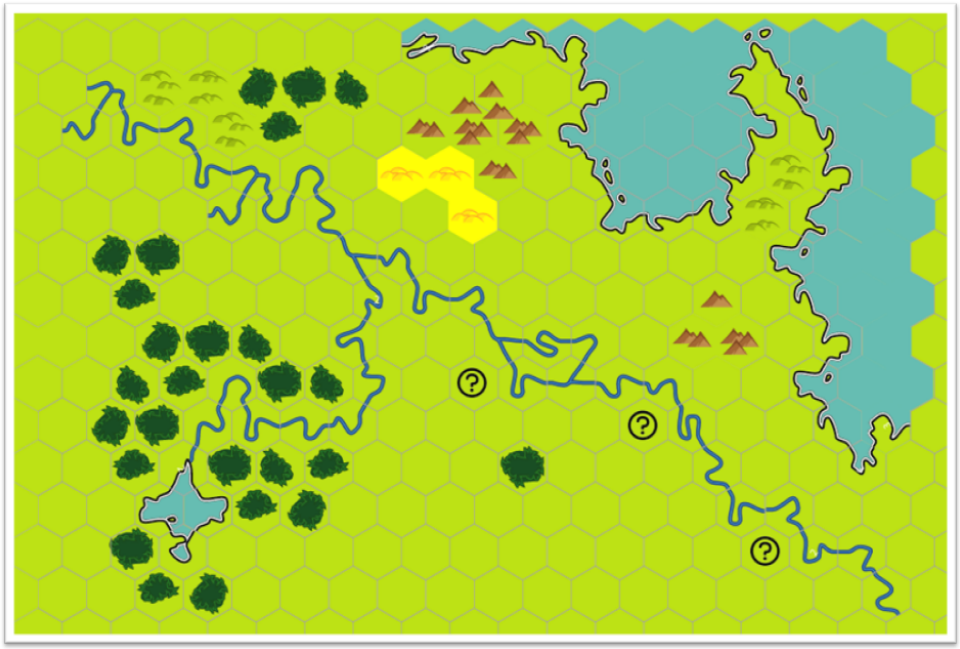


Figura 13 – Gráfico *Burndow* construído como um mapa para aventuras RPG.

Seu enredo foi caracterizado por uma história que envolvia três amigos perdidos em uma floresta e que precisavam encontrar um ponto no qual pudessem atravessar o rio e chegar a outra margem (região abaixo da diagonal do *burndown*).

Para se ter ideia das tarefas realizadas ao longo deste piloto, apresenta-se um conjunto de tarefas de um determinado dia do *sprint*. Nesse dia em específico, esperava-se que a equipe conseguisse concluir as seguintes tarefas:

- Corrigir um erro presente em um arquivo de código fonte de nome Excepcional;
- Solucionar instabilidade da *feature* Estação;
- Realizar operação *merge*[1] com o *branch* principal.

Também a título de ilustração, apresenta-se um trecho do capítulo enviado para os desenvolvedores da equipe, visando ilustrar a relação do enredo criado com as tarefas do *sprint*.

"*...O medo era tanto que correram durante alguns minutos no meio da floresta até que de repente avistaram uma placa fincada no chão que os fizeram parar. A placa continha algo escrito. Daniel encontrou forças, apesar de sua garganta seca, e leu a placa com dificuldades:*

- Aventureiro, 5km a frente, além da floresta, encontram-se as Grandes Montanhas de Weday. Aventurar-se por entre tais montanhas pode custar-lhe nunca mais voltar pra

[1] a operação *merge* é caracterizada pela mescla de versões diferentes de um artefato de um mesmo repositório de controle de versão

casa. Para evitá-las e ter mais chances atravessando o Deserto Amarelo você terá que realizar três tarefas em um dia:

- *Aniquilar um inseto excepcional.*
- *Estabilizar as estações do ano.*
- *Oferecer um merge ao deus Branch.*

Um outro elemento utilizado foi um quadro de pontuação. A Figura 14 exibe o quadro com avatar e pontuações (*XP, Level*) dos jogadores. Durante a experiência, esse quadro ficou sempre exposto na mesa de trabalho da equipe e registrou pontos ganhos de cada desenvolvedor de acordo com a quantidade de pontos de história estimados para cada tarefa.

A pontuação *XP* (do inglês *eXperience Points*, pontos de experiência) é uma unidade de medida usada em muitos jogos RPG para quantificar a progressão de um personagem durante o jogo. Os pontos de experiência são geralmente concedidos para a conclusão de missões e superação de obstáculos e adversários (GILSDORF, 2010).

Na abordagem, a pontuação XP possui duas trilhas de pontuação: uma inferior que era atualizada sempre que um desenvolvedor iniciava uma nova tarefa e uma superior que era atualizada imediatamente após a conclusão da tarefa.

O *Level* era incrementado à medida que o desenvolvedor alcançava certas pontuações de *XP* na trilha superior: 1xp, 5xp, 15xp, 45px. Esses valores foram definidos de forma arbitrária como estimativa de incremento de dificuldade.

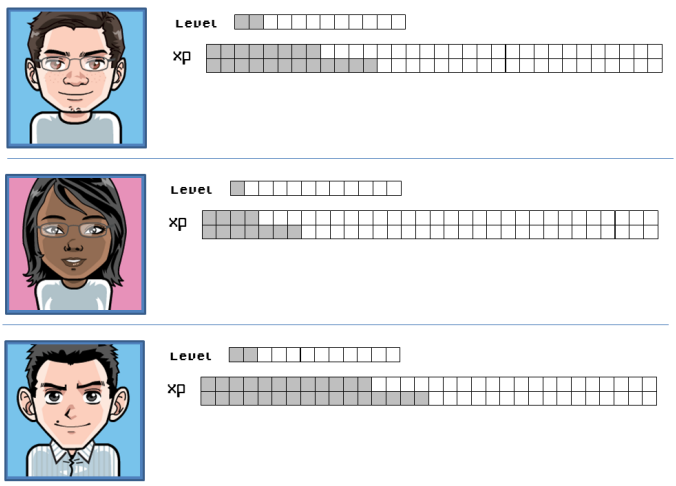


Figura 14 – Quadro de pontuação dos desenvolvedores. As pontuações representam avanços de início e conclusão de tarefas por cada desenvolvedor de acordo com os pontos de história estimados por cada tarefa.

11.2 Análise

Ao final da utilização da metáfora do RPG no desenvolvimento de software foi realizada uma entrevista informa com todos os integrantes a equipe de forma coletiva. Como pontos positivos, os desenvolvedores afirmaram se sentir envolvidos pela história criada, sobretudo por fazerem parte dela, a ponto disso influenciar positivamente no engajamento na execução das tarefas. Como ponto negativo, a entrevista indicou que o quadro de pontuação não causou o efeito desejado e pouco gerou engajamento da equipe, mesmo após os desenvolvedores reconhecerem seus avatares por conta da semelhança caricata com cada um deles. Um dos possíveis fatores desse baixo impacto apontado pela equipe foi a pequena quantidade de jogadores envolvidos (apenas três), agravado pelo fato de um dos desenvolvedores estarem sujeitos a uma carga horária de trabalho menor quando comparada às jornada dos outros desenvolvedores. A entrevista realizada pela equipe não seguiu nenhum roteiro previamente definido. De forma deferente, o estudo de caso apresentado no capítulo 12 utilizou um roteiro pré-definido para a entrevista e foi realizada de forma individual com cada um dos participantes.

Pode-se observar que os desenvolvedores incorporaram a figura dos personagem, de modo a, em vários momentos, referenciar as tarefas do *sprint* utilizando as analogias apresentadas, ao invés dos nomes reais, e realizar citações aos acontecimentos como: "*Ontem tivemos que atravessar o deserto, precisamos nos reaproximar do rio hoje!*".

A partir dessa observações, suspeita-se que o elemento história criou o engajamento desejado dos desenvolvedores com as atividades assumidas como compromisso no planejamento do *sprint*.

Um ponto negativo dessa avaliação é que a pessoa que assumiu o papel de GM foi um dos desenvolvedores da equipe, fazendo com que esse desenvolvedor não participasse do RPG como um jogador. Outro ponto negativo observado foi o relato do GM, que afirmou que a condução do jogo lhe exigiu considerável dedicação, para que fosse mantida uma história engajante e divertida.

Considerações finais

Este capítulo apresentou uma avaliação preliminar sobre a introdução da metáfora de um RPG nas tarefas de desenvolvimento de uma equipe real. Apesar de não seguir um protocolo formal de pesquisa, a avaliação auxiliou na preparação e condução do estudo de caso apresentado no próximo capítulo. Foram identificados pontos positivos e negativos, além de desafios a se lidar com introdução de gamificação no desenvolvimento de software. Esses pontos foram explorados para a definição do estudo de caso que será apresentado no próximo capítulo.

12 Estudo de caso

Introdução

Alguns trabalhos abordados no Capítulo 7 apresentam o uso de mecânicas de jogos em alguns aspectos do processo de desenvolvimento de *software*, mas sem apresentar estudos que tenham como objetivo indicar os efeitos ocasionados pela introdução de gamificação nesse contexto, como por exemplo, a alteração de produtividade, foco, engajamento e a recepção desse tipo de cultura pelos desenvolvedores.

Neste capítulo será descrito o estudo de caso realizado com o intuito de verificar os efeitos da gamificação no processo de desenvolvimento de *software*.

As próximas seções detalham o estudo de caso usando a seguinte organização: a Seção 12.1 apresenta o plano do estudo de caso, posteriormente é descrita a condução do estudo, na Seção 12.3, e a Seção 12.4 apresenta seus resultados e discussão.

12.1 Planejamento

Esta seção apresenta o plano do estudo de caso, destacando seus objetivos, questões, contexto, participantes e procedimentos.

12.1.1 Objetivo e questões da pesquisa

O objetivo principal deste estudo é analisar o impacto da aplicação de mecânicas de jogos nas atividades de equipes de desenvolvimento de *software*. O objetivo é refinado em um grupo de questões de pesquisa que, ao serem respondidas, demonstram se o estudo de caso alcançou seu objetivo. As questões de pesquisa elencadas (QP) para o estudo foram:

- QP1 - A produtividade de uma equipe de desenvolvimento de *software* é influenciada quando são inseridas mecânicas de jogos nas suas atividades?
- QP2 - Quais os efeitos positivos da proposta?
- QP3 - Quais os efeitos negativos da proposta?
- QP4 - Como é a recepção da gamificação por parte dos desenvolvedores e líderes de equipe?

12.1.2 Proposições e hipóteses

Este estudo de caso tem como objetivo, em grande parte, observar como a produtividade de desenvolvedores pode ser afetada pela gamificação de seu processo de desenvolvimento. Em outras palavras, tentar estudar a seguinte questão:

A produtividade de uma equipe de desenvolvimento de *software* é alterada quando são inseridas mecânicas de jogos nas atividades e processo de desenvolvimento da equipe?

Há duas hipóteses diretas que surgem a partir da proposição apresentada. A primeira, hipótese nula, indica que a gamificação não gera efeitos sobre a produtividade, e a segunda, a hipótese alternativa, afirma que a gamificação gera efeitos na produtividade. As hipóteses são descritas a seguir:

- Hipótese nula, H_0produtividade: **não há alterações** na produtividade de uma equipe de desenvolvimento de *software*, quando praticada a gamificação no processo de desenvolvimento utilizado pela equipe. H_0 produtividade: Produtividade (com gamificação) = Produtividade(sem gamificação).
- Hipótese alternativa: **há alterações** de produtividade de uma equipe de desenvolvimento quando praticada a gamificação nas atividades e processo de desenvolvimento da equipe. H_1produtividade: Produtividade(com gamificação) $\neq$ Produtividade(sem gamificação)

Apesar das proposição e hipóteses considerarem que a produtividade pode sofrer ou não alterações, o real efeito esperado é que a produtividade sofra alterações positivas, ou seja, aumente, indicando nesse caso um primeiro indício da gamificação como uma boa ferramenta para o desenvolvimento de software.

12.1.3 Seleção de variáveis

Para considerar especificamente a influência da gamificação na produtividade das equipes, há de se identificar as variáveis independentes envolvidas. Variáveis independentes são fatores que quando alterados acredita-se influenciar a variável observada (dependente). Nesse contexto, a variável dependente é a produtividade e as independentes (fatores que podem influenciá-la) são:

- a aplicação de gamificação no processo de desenvolvimento de *software*;
- a experiência e maturidade atuais da equipe de desenvolvimento;
- as ferramentas utilizadas no ambiente de desenvolvimento, tais como a IDE, linguagens de programação e gerenciadores de tarefas;

- o escopo do projeto de *software* desenvolvido, que define o potencial mercadológico e relevância do produto gerado;
- o tamanho e distribuição geográfica da equipe;
- o processo de *software* praticado pela equipe;
- a própria produtividade atual da equipe;
- a cultura organizacional na qual a equipe de desenvolvimento está inserida, incluindo desde rigorosidade com cronogramas e acompanhamento de metas a políticas de promoções;
- a insalubridade física do ambiente onde a equipe trabalha;
- a relação interpessoal entre os membros da equipe.

Este grupo de variáveis independentes, a depender do contexto no qual se realizará um estudo de produtividade de equipes de desenvolvimento, pode apresentar mais variáveis, bem como ter diferentes intensidades de relevância.

Dentre as variáveis independentes relacionadas, a aplicação de gamificação é a variável independente no qual este estudo de caso trabalhará, enquanto as demais não serão 'estimuladas' ou alteradas pelo estudo. É importante destacar que, como esta pesquisa trabalha com um ambiente real, há pouco controle sobre as demais variáveis.

12.2 Conjectura e unidade de análise

A questões de pesquisa deste estudo estão associadas aos fatores presentes no cotidiano de desenvolvimento de software. Em outras palavras, refere-se ao conjunto de atividades incluindo, por exemplo, as atividades executadas pelos desenvolvedores, a relação entre os desenvolvedores como equipe, com gerentes de projeto e clientes, os projetos em desenvolvimento e produtos de software gerados, além da motivação dos desenvolvedores diante das tarefas diárias.

Por sua natureza (estudo de caso), a pesquisa foi conduzida (*in vivo*), dentro de um ambiente onde há baixo controle, mas alto grau de realismo. A unidade de análise definida foi um grupo real de desenvolvimento do setor privado, formado por profissionais experientes em desenvolvimento. O grupo é responsável simultaneamente pelo desenvolvimento de aplicações, ainda não submetidas ao ambiente de produção, bem como pela manutenção de *software* já em produção. O grupo constitui o setor de desenvolvimento da Infoway [1], uma empresa privada de tecnologia com mais de 18 anos de experiência, que trabalha com

[1] http://www.infoway-pi.com.br

produtos de software para apoio à gestão, especialmente ligados à área da saúde. O grupo adota o Scrum como seu processo de trabalho há mais de 6 anos com papéis PO, SM e equipe definidos, além dos artefatos e cerimônias Scrum já fazerem parte de sua cultura organizacional. Ao todo, o grupo totalizava, no momento deste estudo, 19 desenvolvedores distribuídos em 4 equipes multidisciplinares, sob a liderança de três gerentes de projeto.

12.2.0.1 Métodos de coleta dados

Foram utilizados diferentes fontes de dados para a realização da coleta de dados antes e depois da execução do estudo de caso, listados a seguir:

- **Entrevista com os desenvolvedores**: realizada com cada desenvolvedor com formato semi estruturado. Foi baseada em um roteiro, mas durante a sua realização, a ordem das perguntas foi redefinida de forma a otimizar sua execução. O roteiro é apresentado no Apêndice A. Essa entrevista foi realizada antes do estudo, visando identificar o contexto antes do início da avaliação;

- **Registros de realização de tarefas**: a contabilização da quantidade de tarefas realizadas pelas equipes de desenvolvimento foi obtida a partir das ferramentas de registros de tarefas existentes na empresa, tendo sido obtido dados referentes a períodos anteriores à realização do estudo de caso e os dados obtidos durante a aplicação da gamificação;

- **Repositórios de código de fonte dos software desenvolvidos**: algumas métricas relacionadas à quantidade de alteração de código foram obtidas na empresa. Esses dados também estavam relacionados a períodos anteriores ao estudo de caso, bem como durante a realização da gamificação.

- **Questionário aplicado aos desenvolvedores**: foi criado um questionário com o objetivo de identificar efeitos percebidos pelos desenvolvedores em seu contexto de trabalho, após a realização do estudo, visando com isso caracterizar o sentimento percebido pela equipe com a introdução da gamificação. O questionário é composto por questões objetivas e subjetivas, contemplando aspectos como consequências positivas e negativas percebidas pela realização da gamificação, continuidade da realização da gamificação e percepção de produtividade individual e das equipes. As questões presentes no questionário são apresentadas no Apêndice B

12.2.0.2 Resumo da definição

O estudo de caso tem como objetivo analisar a aplicação de elementos de *game design* nas atividades de uma equipe de desenvolvimento de *software*, com o propósito avaliar seus efeitos em aspectos do contexto de equipes de desenvolvimento, do ponto de

vista e no contexto de desenvolvedores de uma empresa privada de desenvolvimento de *software*.

12.2.1 Validade

Durante a realização de um estudo de caso é importante entender as ameaças que o estudo está submetido e com isso pensar em meios de evitar que o estudo seja considerado inválido. Por conta disso, são discutidos a seguir questões que podem influenciar no estudo e minimizar sua validade.

12.2.1.1 Validade Interna

A validade interna de um estudo de caso está relacionada a fatores que definem se o relacionamento observado entre o tratamento e o resultado é causal e não é consequência da influência de outro fator (WOHLIN et al., 2012).

Um exemplo de ameaça à validade interna deste trabalho está associada ao fato de escopos, prazos, recursos e tecnologias envolvidas nos projetos serem dinâmicos e poderem ser alteradas durante o estudo. Isso significa que uma mudança nos rumos de um projeto pode gerar efeitos nas variáveis observadas que não são associados ao processo de gamificação.

Além disso, cada equipe trabalha em fases diferentes nos seus projetos. Para reduzir essa ameaça, não foram consideradas comparações de medições de produtividade entre equipes, apenas da mesma equipe.

A data de realização do estudo pôde ser influenciada por rumos diferentes da equipe dentro do projeto, como redução da equipe por licenças de saúde ou férias. Para mitigar os efeitos dessa ameaça, as métricas quantitativas de produtividade foram normalizadas para fins de comparação, tentando assim utilizar valores normalizados por membro.

A aplicação do gamificação no processo de desenvolvimento exige competências de *game designer*, de forma que o processo de desenvolvimento torne-se mais interessante. Essa tarefa foi assumida pelo autor desta pesquisa, embora ele não tenha formação nem experiência nessa área de atuação. Por conta disso, a deficiência nessa competência pode ameaçar o sucesso da aplicação da gamificação neste estudo de caso.

Parte dos métodos de obtenção dos será realizada através de entrevistas e questionários com os desenvolvedores integrantes das equipes participantes do estudo de caso. A postura assumida pelos desenvolvedores durante as entrevistas e ao responder os questionários poderá apresentar-se como ameaça no momento que suas respostas assumirem versões atenuadas ou exageradas da realidade. Isso pode ocorrer quando os desenvolvedores, por alterar suas respostas, acreditem estar ajudando na obtenção de melhores resultados na

pesquisa por “coleguismo” com os responsáveis por estar pesquisa. Para reduzir os riscos desta ameaça esse aspecto será abordado e esclarecido, durante cada uma das entrevistas.

12.2.1.2 Validade Externa

A validade externa refere-se às condições que limitam a habilidade de generalizar os resultados do estudo. Da mesma forma que a ameaça à validade anterior, foram inferidas algumas ameaças que podem influenciar no estudo.

A diversidade de organização estrutural, tamanho e experiência das equipes podem impactar em como a aplicação de *técnicas de game design* influenciam a produtividade das equipes de desenvolvimento, evidenciando uma ameaça a validade externa deste estudo. Além da diversidade quanto a equipes, diferentes ferramentas, projetos e culturas organizacionais são outros fatores que podem ser de fundamental importância para expandir as conclusões desse estudo a outros contextos.

12.3 Operação

O estudo de caso iniciou-se com a realização das entrevista com os desenvolvedores. Foram entrevistados 11 de um total de 19 desenvolvedores. A seleção dos entrevistados considerou critérios como: indicação de superiores e líderes de equipe, disponibilidade (dois desenvolvedores estavam em regime de férias, um desenvolvedor enfrentou questões de saúde que provocaram uma licença de mais de três semanas) e volume de entrevistados de cada equipe.

As entrevistas foram realizadas com o auxílio do roteiro apresentado no Apêndice A. Dessa forma, cada desenvolvedor entrevistado foi interrogado sobre a sua percepção de produtividade da empresa como um todo, da sua equipe e, por fim, da sua própria produtividade. Além disso, foram questionados os três principais fatores determinantes para tais patamares de produtividade. Foram realizadas em um período de duas semanas dentro do ambiente da empresa, mas de forma reservada com cada um dos 11 entrevistados. As entrevistas duraram entre trinta minutos e uma hora e tiveram o áudio gravado para posterior análise. A partir de cada entrevista foram extraídos as percepções de cada desenvolvedor quanto à sua própria produtividade, da equipe e da empresa com um todo, além da identificação dos aspectos considerados definidores de produtividade.

A ferramenta descrita na Seção 9.4 foi instalada no ambiente da empresa e operou neste ambiente durante dez semanas. Nesse período, foram realizadas conversas informais com alguns desenvolvedores e líderes de equipe para ajuste de funcionalidade e manutenção das mecânicas. Como exemplo, a manutenção incluiu: a seleção das tarefas a ser marcadas como como tarefas premiadas; amadurecimento da redação das mensagens de *feedback*; correções e balanceamentos no regras de cálculo de pontuação de xp; alteração nos valores de referência *xp* para alcance de próximo *level*; refinamentos visuais de ícones, fontes e cores; e vulnerabilidades que permitiam trapaças.

Optou-se por não apresentar o modo de funcionamento das mecânicas, com o intuito de "dificultar" ou "retardar" a prática de trapaça pelos jogadores. Um aspecto interessante é que durante o estudo de caso, por iniciativa de alguns dos desenvolvedores participantes, algumas vulnerabilidade que permitiam trapaças forma identificadas e relatadas para fins de correção.

É importante ressaltar a dificuldade e o cuidado necessário nesse estudo, por conta de se tratar de equipes reais em ambiente reais de trabalho. A ferramenta de gestão de tarefas está inserida um ambiente corporativo, com operacionalização que afeta clientes e negócios reais em diferentes estados do Brasil. Por conta disso, o escopo e natureza da pesquisa foi apresentado e recebeu aval de um dos diretores da empresa, além do seu gerente de tecnologia, que era responsável pelas equipes de desenvolvimento.

Após dez semanas, foram realizadas contabilizações no repositório de código dos

software desenvolvidos e no registro de realizações de tarefas referente às medidas de produtividade apresentadas no Capítulo 3. Além disso, foi aplicado um questionário *on line* para todos os desenvolvedores. As questões utilizadas no questionário são apresentadas no Apêndice B

A análise dos dados foi realizada da seguinte forma: dados coletados sobre produtividade na entrevista foram comparados com informações do questionário. Essa comparação confrontou a percepção de produtividade pelos desenvolvedores antes e após a realização da gamificação, de modo a evidenciar se, de acordo com os desenvolvedores, fatores definidos como decisivos para a produtividade foram afetados durante a gamificação.

Dados coletados dos registros de realização de tarefas e repositórios de código foram também comparados quantitativamente para identificação de alterações no volume de tarefas realizadas e volume de código incrementado ou alterado antes e durante o período de aplicação da gamificação. Essa análise contou com fundamental contribuição de desenvolvedores das equipes participante do projeto, por conta da quantidade de ruído presentes nos dados.

12.4 Resultados e discussão

A partir da compilação das entrevistas e questionários, foram identificadas as percepções de produtividade própria, das respectivas equipes e da empresa como um todo. A percepção foi coletada em uma escala numérica de zero a dez, onde zero significa uma produtividade nula, na qual a equipe ou desenvolvedor não conseguem realizar quaisquer atividades, e dez representa um produtividade fantástica, convencionada durante a entrevista e questionário como “perfeita” e sem possibilidade ou necessidade de melhoria.

Os valores da produtividades extraídos das entrevistas e questionário são listados na Tabela 6. Para cada desenvolvedor são apresentados uma nota para a percepção de produtividade da empresa como um todo, para a equipe onde o desenvolvedor está inserido e outra para a própria produtividade, tanto registradas antes da aplicação da gamificação, “Pré”, como após a aplicação, “Pós”.

As informações das colunas “Pré” da tabela foram coletadas a partir das entrevista e as das colunas “Pós” a partir dos questionários. As células das colunas “Pré”, que apresentam valores “-”, são para os desenvolvedores não entrevistados (desenvolvedores 7 e 14, por exemplo). Já para as colunas “Pós”, as células que apresentam valores “-” representam desenvolvedores que não realizaram o questionário por alguma indisponibilidade, dado que o questionário foi submetido *online* e a sua solicitação de realização foi feita presencialmente a todos os desenvolvedores. Dessa forma, dos dezenove desenvolvedores participantes do estudo de caso, catorze foram entrevistados (74%), quinze responderam o questionário (79%) e onze (58%) tanto foram entrevistados como responderam ao

Tabela 6 – Listagem das notas de percepção de produtividades extraídos antes e após a gamificação para a empresa, equipe e respectivo desenvolvedor.

		Prod Empresa		**Prod Equipe**		**Prod Própria**	
Desenvolvedor	**Equipe**	**Pré**	**Pós**	**Pré**	**Pós**	**Pré**	**Pós**
1	A	5	-	5	-	6	-
2	A	4	6	3	5	3	5
3	A	7	8	8	7	8	8
4	B	6	8	6	8	7,5	7
5	B	7	-	9	-	7	-
6	B	7	8	7	8	7	8
7	B	-	7	-	7	-	7
8	B	7	-	8	-	8	-
9	B	7	9	8	9	7	8
10	C	7	7	8	8	8	8
11	C	6	7	7	8	7	7
12	C	-	-	-	-	-	-
13	C	6	6	4	7	5	9
14	D	-	7	-	8	-	-
15	D	7	7	7	7	7	8
16	D	-	7	-	8	-	-
17	D	7	7	7,5	8	7	7
18	D	8,5	8	8,5	9	8,5	9
19	D	-	7	-	7	-	-

questionário. Um ponto a se destacar é que desses onze, apenas o Desenvolvedor 3, da Equipe A, e o Desenvolvedor 19, da Equipe D, apresentaram a segunda nota (Pós) de produtividade para equipe e para toda a empresa, respectivamente, inferiores às primeiras notas, enquanto os demais apresentaram a segunda nota de produtividade superior ou igual a primeira nas três esferas, da empresa, da equipe e própria.

A Tabela 7 apresenta as médias das notas de produtividade apresentadas na Tabela 6 por equipe e geral. Apenas a equipe D apresentou a média "Pós" para a segunda nota inferior à primeira para produtividade da empresa, enquanto as demais as equipes apresentaram crescimento das notas médias para todas as classificações. Na comparação geral, pode-se afirmar que a percepção dos desenvolvedores, quanto às produtividades da empresa, das equipes, e pessoais foram, de maneira geral, de melhora.

A compilação das entrevistas resultou também na listagem dos fatores que influenciavam a produtividade existente antes da realização do estudo, ou seja, a produtividade percebida pela equipe. Cada um dos catorze desenvolvedores entrevistados apresentou três aspectos que juntos totalizaram onze diferentes fatores de influência. Os aspectos estão listados na Tabela 8, acompanhados do percentuais de indicação. Ou seja, os fatores listados nessa tabela eram percebidos como fatos que impediam as equipes de terem uma

Tabela 7 – Listagem das notas médias de percepção de produtividades por equipe extraídas antes e após a gamificação para a empresa, equipe e respectivos desenvolvedores.

	Geral		**Equipe**		**Própria**	
Equipe	Pré	Pós	Pré	Pós	Pré	Pós
A	5,33	7,00	5,33	6,00	5,67	6,50
B	6,80	8,00	7,60	8,00	7,30	7,50
C	6,33	6,67	6,33	7,67	6,67	8,00
D	7,50	7,17	7,67	7,83	7,50	8,00
Média Geral	6,54	7,27	6,86	7,60	6,86	7,58

produtividade melhor antes do estudo.

Tabela 8 – Fatores definidos como determinísticos para os patamares de produtividade apontados pelos desenvolvedores antes da aplicação de gamificação.

Fatores	**Frequência**	**%**
Processo de software mal estruturado	8	19%
Mau planejamento e má organização de tarefas	7	17%
Tecnologia adotada (*frameworks*) obsoleta	6	14%
Dispersão/falta de foco	4	10%
Baixo conhecimento técnico dos desenvolvedores	4	10%
Falta de motivação	3	7%
Código legado de difícil manutenção	3	7%
Precariedade de integração de componentes de software	2	5%
Falta de liderança	2	5%
Repetitividade/Similaridade de tarefas	2	5%
Alto volume de falhas após liberação	1	2%

A aplicação do questionário coletou informações sobre influências positivas e negativas percebidas pelos desenvolvedores devido a introdução da gamificação. Sobre efeitos positivos, apenas dois desenvolvedores afirmaram não notar alguma consequência. As percepções positivas apontadas pelos outros desenvolvedores são listadas na Tabela 9.

A grande maioria dos catorze desenvolvedores que respondeu o questionário afirmou que não notou influências negativas causadas pela aplicação da gamificação. No entanto, um dos desenvolvedores afirmou que percebeu o surgimento de especulações sobre implicações de maus resultados no “jogo”, o que como consequência fomentaria a trapaça e uma corrida na realização de atividades em detrimento da qualidade do trabalho.

É interessante destacar alguns pontos apresentados na Tabela 8 e as influências listadas na Tabela 9. Um primeiro ponto de destaque é que, de acordo com quatro desenvolvedores, notou-se a realização de melhores planejamento, ao passo que “mau planejamento e má organização de tarefas” foi apresentado como o segundo fator que mais afetava negativamente a produtividade antes do estudo. As influências positivas

Tabela 9 – Influências positivas ao ambiente de trabalho identificadas pelos desenvolvedores após da aplicação de gamificação.

Influência positiva	**Frequência**
Competição saudável	5
Maior "pressa" para a conclusão da tarefas	5
Melhor planejamento (melhores organização, segmentação e detalhamento de tarefas)	4
Melhor compreensão de progresso na realização de tarefas	3
Nenhuma	2
Desejo de compreensão das tarefas de todos os membros da equipe	1
Incentivo ao uso da ferramenta de gestão de tarefas	1

coletadas a partir do questionário se confirmaram com aspectos identificados conversas informais durante a condução da pesquisas. Isso reforça a redução dos riscos de "falsas" influências negativas apresentadas com o intuito de gerar bons resultados por coleguismo aos pesquisadores. Outro ponto a destacar é que apenas dois desenvolvedores não notaram quaisquer influências positivas.

Sobre a continuidade da gamificação, todos os desenvolvedores que responderam o questionário foram unânimes ao afirmar que a prática deveria continuar. As respostas sobre a continuidade do uso da gamificação são apresentadas na Tabela 10.

Tabela 10 – Sumarização das respostas quanto a continuidade de prática de gamificação no cotidiano de desenvolvimento dos desenvolvedores participantes do estudo.

Respostas	**Frequência**
Acho que deveria continuar, é bem legal isso no dia a dia.	5
Acho que deveria continuar, mas com correções e ajustes.	8
Acho legal que permaneça, mas com mecanismos completamente diferentes.	1
Tanto faz, pois de nada muda o nosso modo de trabalho.	0
Deveria ser removida, pois mais atrapalha do que ajuda.	0

Os resultados sobre o volume de tarefas realizadas, extraídos da ferramenta de registro tarefas da empresa, consideraram registros de um intervalo de vinte semanas, sendo dez semanas imediatamente anteriores ao período da gamificação e dez durante a gamificação. Foram contabilizadas ao todo 1272 tarefas sendo:

- 122 tarefas da Equipe A;
- 441 tarefas da Equipe B;
- 523 tarefas da Equipe C;

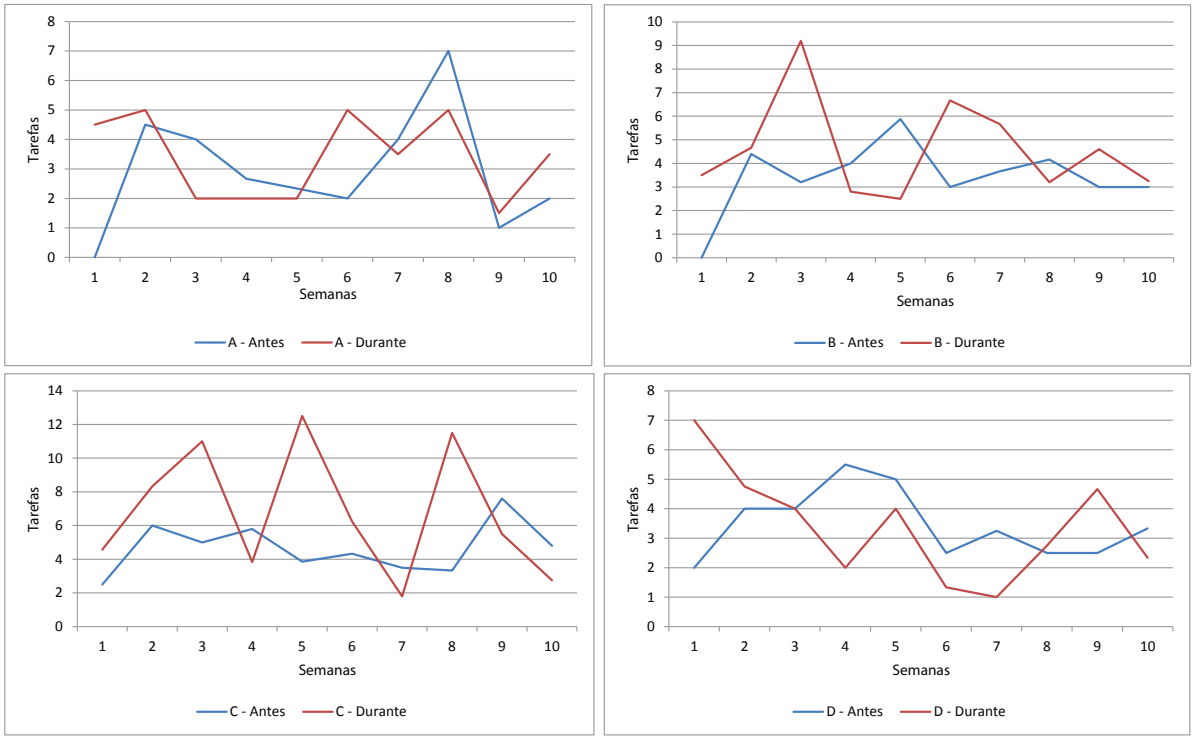


Figura 15 – *Implicitly Estimated Velocity* médio para períodos semanais de cada equipe participante do estudo de caso.

- 186 tarefas da Equipe D;

A Figura 15 apresenta quatro gráficos comparativos da evolução da métrica *Implicitly Estimated Velocity* (Capítulo 3) para as quatro equipes, A, B, C e D. Para a redução de ruídos, por conta de alterações na quantidade de integrantes, a métrica foi racionalizada pelo tamanho da equipe. Desse modo, a informação apresentada em cada gráfico representa a evolução do *Implicitly Estimated Velocity* médio, por desenvolvedor, em cada semana. Apesar da racionalização, uma simples análise visual dos gráficos evidencia a ausência de tendências ou padrões em quaisquer um dos gráficos. Alguns integrantes das equipes foram consultados e todos reconheceram a "imprevisibilidade" de comportamento em uma análise utilizando unidades semanais de tempo. Por conta da duração do período de aplicação da gamificação, dez semanas, uma análise com unidades de tempo maiores se tornou inviável.

A métrica *Velocity* (Capítulo 3) apresentou-se impraticável de ser calculada devido ao fato da cultura de estimativa de custo ou duração de tarefas não estar consolidada como prática das equipes participantes. Apenas 63% das tarefas da equipe A possuíam registro de estimativa, 50% da equipe B, apenas 11% da equipe C e 30% da equipe D.

A partir da análise dos repositórios de código, foram calculas as evoluções da métrica *Hits of Code (HoC)*(Capítulo 3), para as quatro equipes, referente aos mesmo períodos utilizados para a métrica anterior, *Implicitly Estimated Velocity*. Da mesma maneira, a análise visual da evolução da HoC não permite a identificação de tendência de

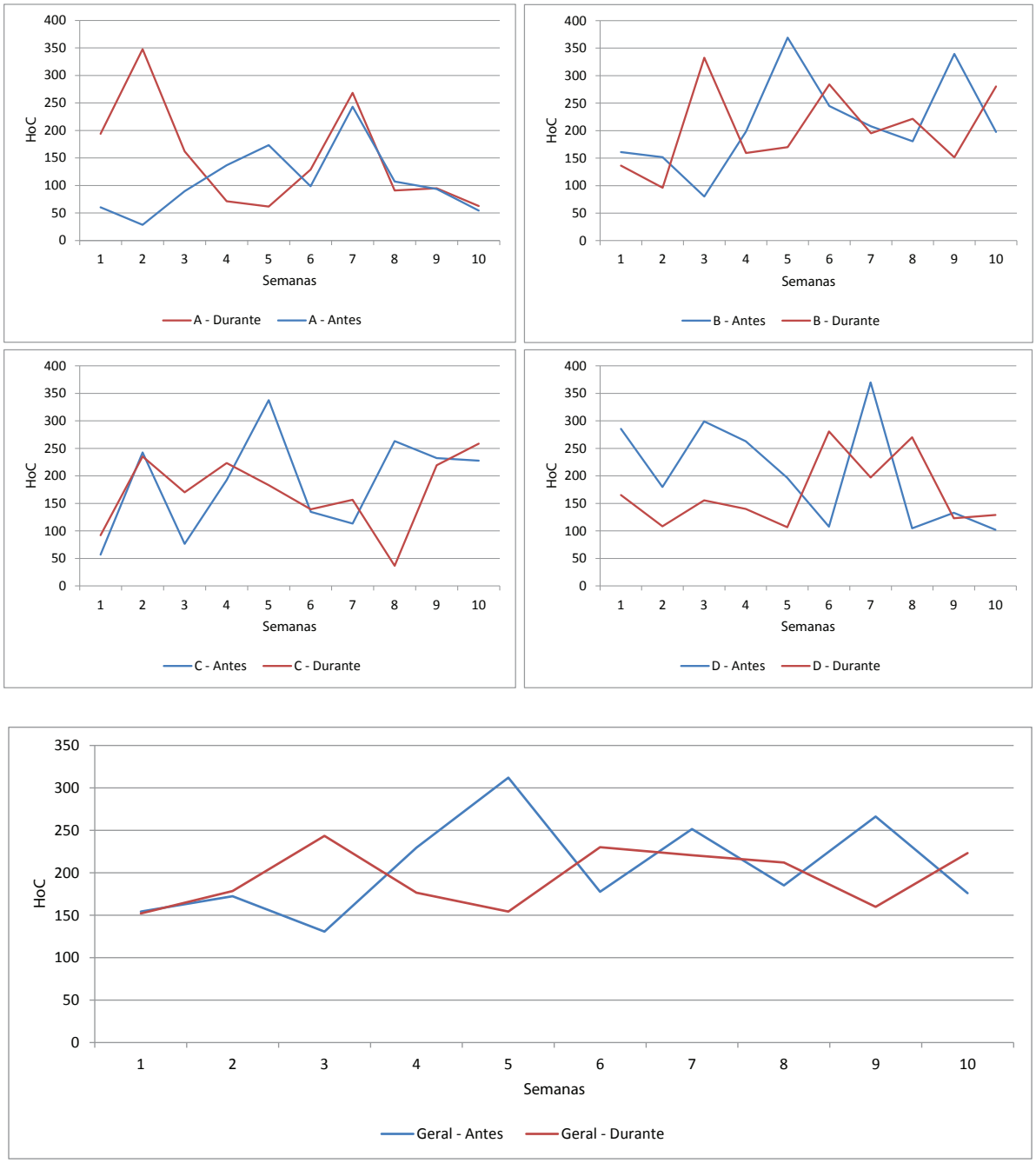


Figura 16 – *Comparativo da evolução de HoC (Hits of Code)* médio entre as dez semanas anteriores(azul) e durante(vermelho) a aplicação de gamificação para as quatro equipes e geral da empresa.

alteração crescente ou decrescente. Ao serem consultados, os desenvolvedores das equipes afirmaram que isso condiz com o cotidiano de trabalho: "*Há semanas em que trabalhamos mais em planejamento e requisitos e pouco lidamos com alteração de código. E isso não é um comportamento regular*".

Os resultados deste estudo de caso mostraram evidências objetivas de melhorias apontadas pelos próprios desenvolvedores participantes do estudo de caso. Informações coletadas em entrevistas e questionários mostraram uma aumento positivo, mas não forte, na percepção dos desenvolvedores quanto às produtividades pessoal e das respectivas equipes. Esse aspecto ataca a questão levantada pela QP1. Assim, podemos afirmar que a produtividade de uma equipe é influenciada pela mecânica de jogos, embora as evidências disso tenham sido fracas, a percepção foi de quase todos os participantes.

Benefícios como maior engajamento na realização de tarefas, realização de planejamentos de iteração mais claros e melhor elaborados e maior lucidez dos desenvolvedores quanto ao andamento de realização de tarefas de iteração, são alguns dos efeitos positivos identificados por conta da aplicação da gamificação. O indício desses benefícios reponde a QP2, que questiona quais eram os efeitos positivos dessa introdução.

A especulação quanto às consequência de mau desempenho no "jogo" foi identificado como um efeito negativo, por estimular comportamentos indesejados, tais como a conclusão de atividades a qualquer custo, em detrimento da qualidade. Esse aspecto responde a QP3, que trata especificamente dos efeitos negativos da proposta.

Os resultados apontaram para a aceitação e desejo de continuação da prática gamificação, ainda que com alterações, por todos os desenvolvedores entrevistados. Essa evidência de aceitação responde positivamente a QP4, que questionava a recepção à abordagem. De um modo geral, ela foi muito bem recebida e foi estimulada a continuar.

Considerações Finais

Este capítulo apresentou o estudo de caso realizado com o objetivo de avaliar a introdução de gamificação no desenvolvimento de software, especialmente nas questões relacionadas a um aumento de produtividade. De um modo geral, a introdução gerou uma percepção positiva junto aos desenvolvedores, além de ter sido estimulada a permanecer como prática dentro da empresa. Essa resultado é bastante estimulante para a pesquisa e denota que muito ainda pode ser feito nesse sentido.

13 Conclusão

A Engenharia de *Software* tem mais de 40 anos de existência. Há avanços em muitas direções, mas ainda há pesquisas e elaboração de ferramentas para resolver os mesmos problemas registrados nesses 40 anos, como projetos com estouro do orçamento, não cumprimento de prazo e não atendimento aos requisitos definidos.

Como mencionado por Brooks (BROOKS JR., 1987), não há bala de prata para os problemas relacionados ao desenvolvimento de *software*, no entanto, é necessário transformar essa atividade em algo que crie engajamento dos desenvolvedores de forma mais eficaz.

A principal contribuição deste trabalho foi mostrar que a teoria de *game design* pode ser aplicada a problemas importantes do mundo real, como a engenharia de *software*, de forma positiva e gerando uma percepção de melhoria para as equipes associadas. Outras contribuições deste trabalho são:

- Uma proposta de gamificação para a engenharia de *software*. Mostrou-se que técnicas de *game design* podem ser aplicadas ao contexto de desenvolvimento de software, especialmente no processo Scrum. A introdução da técnica permite que o desenvolvimento, seja guiado pelo Scrum ou por outro processo, possa ser transformado em uma tarefa mais engajante, como um jogo;

- Um mapeamento de mecânicas de jogos aos conceitos de desenvolvimento de *software*. Foram apresentados vários *achievements* que podem ser incorporados ao Scrum, visando estimular sua adoção e execução;

- Uma avaliação retrospectiva da abordagem para a proposta com dados de uma empresa real de desenvolvimento de *software*. Foi realizada uma análise dos *achievements* para uma empresa. Os resultados da análise mostraram que os *achievements* propostos poderiam ser uma métrica interessante para medir o desempenho de equipes de desenvolvimento, além de possibilitarem um estímulo à competição entre os desenvolvedores. O *feedback* obtido a partir da equipe foi encorajador, por conta dos desenvolvedores da empresa utilizada na avaliação terem ficado entusiasmados com a ideia de incorporação técnicas de *game design* em suas atividades.

- A proposta de especificação de funcionalidade para uma ferramenta de gestão de tarefas que incorpora elementos de *game design*. A ferramenta proposta inclui elementos como o *feedback* imediato, inferência de classes e apresentação de *achievements* e perfis dos jogadores.

- Um estudo de caso que analisou os efeitos da aplicação da gamificação no cotidiano de desenvolvimento de quatro equipes de desenvolvimento, envolvendo 19 desenvolvedores em uma empresa real. Apesar das métricas coletadas da base de código e registro de tarefas não terem evidenciado os efeitos positivos da gamificação na produtividade da equipe, a percepção e opinião dos desenvolvedores participantes do estudo apontaram para benefícios diretos no cotidiano de desenvolvimento.

13.1 Limitações do trabalho

Alguns obstáculos enfrentados durante o trabalho, em especial no estudo de caso apresentado no Capítulo 12, foram aspectos limitantes de algumas conclusões.

A condução do estudo de caso, no que se refere a gamificação do cotidiano de desenvolvimento da empresa participante, foi realizada pela equipe de pesquisa que possuía pouca experiência na criação e manutenção de jogos e ambientes “gamificados", podendo reduzir os efeitos positivos ou negativos de sua aplicação. Além disso, apesar da colaboração de pessoas da empresa participante da pesquisa, a condução da gamificação foi realizada somente pelos pesquisadores deste trabalho, que não viviam o cotidiano da empresa, além de não estarem presentes na empresa durante todo o período do estudo de caso.

A duração do período de aplicação de gamificação comprometeu a análise dos resultados de produtividade das equipes de desenvolvimento, evidenciando a necessidade e oportunidade de realização de estudo similar de mais longa duração.

O não formalismo do registro de prescrições Scrum praticadas na ferramenta de gestão de projetos da empresa, limitou o desenvolvimento da ferramenta utilizada para a realização do estudo de caso descrita na Seção 9.4. Isso evitou a avaliação plena das proposta deste trabalho, uma vez que os *achievements* desenvolvidos dependiam diretamente de métricas provenientes do Scrum.

Apesar do mapeamento entre mecânicas de *game design* e desenvolvimento de software utilizar o Scrum como metodologia de desenvolvimento, há muitos outros aspectos e disciplinas que poderiam ser abordados, tais como qualidade de software, envolvendo desde incidência de falhas e acompanhamento de períodos de indisponibilidade, complexidade de código gerado e satisfação de clientes.

13.2 Trabalhos futuros

Os aspectos apresentados como críticas ao trabalho, na seção anterior, representam diretamente oportunidade de novos trabalhos. A realização de um estudo similar, com maior duração, pode trazer resultados mais fortes para as conclusões sobre questões de pesquisa levantadas. Além disso, a inclusão de outros aspectos do desenvolvimento que

incluam outras fases e profissionais do processo de software, desde designers a analistas de negócios, ou equipes distribuídas geograficamente, podem evidenciar questões, efeitos e desafios não identificados neste trabalho.

13.3 Publicações

A realização deste trabalho gerou algumas publicações que foram importantes para os seus autores, uma vez que funcionaram como mecanismo de *feedback* para seus participantes. As publicações obtidas foram:

- Passos, E. B. ; Medeiros, D. B. ; Neto, Pedro Santos ; Clua, E. W. G. . Turning Real-World Software Development into a Game. In: Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), 2011, Salvador. Anais do X Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), 2011. Esse trabalho apresentou como contribuições um mapeamento de mecânicas de game design ao processo de desenvolvimento de software iterativo, propostas de mecânicas e suas avaliações com base em registro histórico acerca da realização de tarefas de equipes ágeis de desenvolvimento.

- Medeiros, D. B. ; Passos, E. B. ; ARAUJO, W. ; Neto, Pedro de Alcântara dos Santos. Working and Playing with Scrum. International Journal of Software Engineering and Knowledge Engineering, 2015. Esse artigo foi premiado como um dos melhores da conferência e foi desenvolvido como continuidade da pesquisa apresentada na publicação anterior com enfoque no processo Scrum.

- Passos, E. B. ; Medeiros, D. B. ; ARAUJO, W. ; SANTOS NETO, P. . Working and Playing with SCRUM. In: The 24th International Conference on Software Engineering and Knowledge Engineering, 2012, Redwood City. Proceedings of the 24th International Conference on Software Engineering and Knowledge Engineering, 2012. Essa publicação é uma versão estendida do trabalho anterior submetida por convite pela premiação de melhores artigos.

Uma contribuição maior deste trabalho é mostrar a aplicabilidade do conhecimento amadurecido da indústria de jogos em outras atividades, a partir da proposta e avaliações em estudos de caso. Acredita-se que isso possa contribuir com as soluções e amadurecimento de problemas no desenvolvimento de software, permitindo assim gerar grande valor para a indústria de software.

Referências

ABRAHAMSSON, P. et al. *Agile software development methods: Review and analysis*. [S.l.]: VTT Finland, 2002. Citado na página 19.

ALVAREZ, J.; DJAOUTI, D. An introduction to serious game definitions and concepts. *Serious Games & Simulation for Risks Management*, p. 11, 2011. Citado na página 6.

BAKER, A.; NAVARRO, E. O.; HOEK, A. V. D. Problems and programmers: An educational software engineering card game. In: *In ICSE O03: Proceedings of the 25th International Conference on Software Engineering*. [S.l.]: IEEE Computer Society, 2003. p. 614–619. Citado 2 vezes nas páginas 34 e 41.

BAKER, A.; NAVARRO, E. O.; HOEK, A. van der. An experimental card game for teaching software engineering processes. *Journal of Systems and Software*, v. 75, n. 1-2, p. 3 – 16, 2005. ISSN 0164-1212. Software Engineering Education and Training. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0164121204000378>. Citado 3 vezes nas páginas 6, 34 e 41.

BEECHAM, S. et al. Motivation in software engineering: A systematic literature review. *Information and software technology*, Elsevier, v. 50, n. 9, p. 860–878, 2008. Citado 2 vezes nas páginas 6 e 15.

BELL, J. B.; WHALEY, B. *Cheating and deception*. [S.l.]: Transaction Publishers, 1991. Citado na página 30.

BEVILACQUA, A. *Redmine Plugin Extension and Development*. [S.l.]: Packt Publishing Ltd, 2014. Citado na página 60.

BLACKBURN, J. D.; SCUDDER, G. D.; WASSENHOVE, L. N. V. Improving speed and productivity of software development: a global survey of software developers. *Software Engineering, IEEE Transactions on*, IEEE, v. 22, n. 12, p. 875–885, 1996. Citado na página 21.

BOEHM, B. W. Improving software productivity. In: CITESEER. *Computer*. [S.l.], 1987. Citado na página 21.

BOEHM, B. W. et al. *Software engineering economics*. [S.l.]: Prentice-hall Englewood Cliffs (NJ), 1981. Citado na página 15.

BROOKS JR., F. P. No silver bullet essence and accidents of software engineering. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 20, n. 4, p. 10–19, abr. 1987. ISSN 0018-9162. Disponível em: <http://dx.doi.org/10.1109/MC.1987.1663532>. Citado 2 vezes nas páginas 5 e 91.

BUGAYENKO, Y. *Hits-of-Code Instead of SLoC*. 2014. <http://www.yegor256.com/2014/11/14/hits-of-code.html>. Citado na página 22.

CHEN, E. T. The gamification as a resourceful tool to improve work performance. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 473–488. Citado 2 vezes nas páginas 33 e 34.

CHEN, J. Flow in games (and everything else). *Communications of the ACM*, ACM, v. 50, n. 4, p. 31–34, 2007. Citado na página 25.

CLAYPOOL, K.; CLAYPOOL, M. Teaching software engineering through game design. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 37, p. 123–127, June 2005. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/1151954.1067482>. Citado na página 6.

CONSALVO, M. Gaining advantage: How videogame players define and negotiate cheating, changing views: Worlds in play. In: *second annual conference of the Digital Games Research Association*. [S.l.: s.n.], 2005. Citado na página 30.

COOK, D. *What are game mechanics?* 2006. <http://www.lostgarden.com/2006/10/what-are-game-mechanics.html>. Citado na página 25.

COOK, D. *What activities can be turned into games?* 2008. <http://www.lostgarden.com/2008/06/what-actitivies-that-can-be-turned-into.html>. Citado na página 26.

COPIER, M. Connecting worlds. fantasy role-playing games, ritual acts and the magic circle. 2005. Citado na página 31.

CORCORAN, E. *Gaming Education*. 2010. <http://radar.oreilly.com/2010/10/gaming-education.html>. Citado na página 6.

COSTA, C. D. *7 potential pitfalls of gamification*. 2012. <http://www.imediaconnection.com/content/31753.asp>. Citado na página 35.

COSTA, C. D. *Cheating at Candy Crush Saga*. 2012. <http://www.gamasutra.com/blogs/MarcusCarter/20150804/250325/Cheating_at_Candy_Crush_Saga.php>. Citado na página 30.

DANELLI, F. Implementing game design in gamification. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 67–79. Citado na página 35.

DEMARCO, T.; LISTER, T. *Peopleware: productive projects and teams*. [S.l.]: Addison-Wesley, 2013. Citado na página 15.

DERBY, E.; LARSEN, D.; SCHWABER, K. *Agile retrospectives: Making good teams great*. [S.l.]: Pragmatic Bookshelf Raleigh, NC, Dallas, TX, 2006. Citado na página 19.

DETERDING, S. Gamification: designing for motivation. *interactions*, ACM, v. 19, n. 4, p. 14–17, 2012. Citado na página 33.

DETERDING, S. et al. From game design elements to gamefulness: Defining "gamification". In: *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*. New York, NY, USA: ACM, 2011. (MindTrek '11), p. 9–15. ISBN 978-1-4503-0816-8. Disponível em: <http://doi.acm.org/10.1145/2181037.2181040>. Citado na página 33.

DOWNEY, S.; SUTHERLAND, J. Scrum metrics for hyperproductive teams: How they fly like fighter aircraft. In: IEEE. *System Sciences (HICSS), 2013 46th Hawaii International Conference on*. [S.l.], 2013. p. 4870–4878. Citado na página 22.

DUBOIS, D. J.; TAMBURRELLI, G. Understanding gamification mechanisms for software development. In: ACM. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*. [S.l.], 2013. p. 659–662. Citado na página 43.

DYER, R. A conceptual framework for gamification measurement. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 47–66. Citado na página 33.

FALSTEIN, N. *Hits-of-Code Instead of SLoC*. 2004. <http://www.gamasutra.com/view/feature/130573/natural_funativity.php>. Citado 2 vezes nas páginas 13 e 14.

FOWLER, M.; HIGHSMITH, J. *The Agile Manifesto*. 2001. In Software Development, Issue on Agile Methodologies, <http://www.sdmagazine.com>, last accessed on March 8th, 2006. Citado na página 5.

GILSDORF, E. *Fantasy Freaks and Gaming Geeks: An Epic Quest for Reality Among Role Players, Online Gamers, and Other Dwellers of Imaginary Realms*. [S.l.]: Globe Pequot, 2010. Citado na página 75.

GOUVEIA, T. et al. Motivation in software engineering: A systematic review update. In: IET. *Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on*. [S.l.], 2011. p. 154–163. Citado na página 14.

HAMARI, J.; ERANTI, V. Framework for designing and evaluating game achievements. *Proc. DiGRA 2011: Think Design Play*, v. 115, p. 122–134, 2011. Citado na página 28.

HAMARI, J.; KOIVISTO, J.; SARSA, H. Does gamification work?–a literature review of empirical studies on gamification. In: IEEE. *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. [S.l.], 2014. p. 3025–3034. Citado na página 41.

HARVIAINEN, J. T. Live-action, role-playing environments as information systems: an introduction. *Information Research*, v. 12, 2007. Citado na página 31.

HAUGE, J. B. et al. State of the art of serious games for business and industry. In: IEEE. *Concurrent Enterprising (ICE), 2011 17th International Conference on*. [S.l.], 2011. p. 1–8. Citado na página 6.

HENRY, L. Group narration: Power, information, and play in role playing games. *Darkshire. net*, 2003. Citado na página 31.

JACKSON, S. et al. *Gurps: Basic Set: Campaigns*. [S.l.]: Steve Jackson Games, 2004. Citado na página 31.

JOHNSON, P. M.; KOU, H. Automated recognition of test-driven development with zorro. In: IEEE. *Agile Conference (AGILE), 2007*. [S.l.], 2007. p. 15–25. Citado na página 43.

KOHWALTER, T. C.; CLUA, E. W.; MURTA, L. G. Sdm-an educational game for software engineering. In: IEEE. *Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on*. [S.l.], 2011. p. 222–231. Citado na página 41.

KOSTER, R. *Theory of fun for game design*. [S.l.]: "O'Reilly Media, Inc.", 2013. Citado 2 vezes nas páginas 14 e 25.

LESYUK, A. *Mastering Redmine*. [S.l.]: Packt Publishing Ltd, 2013. Citado na página 60.

LINDLEY, C. Narrative structure in trans-reality role-playing games: Integrating story construction from live action, table top and computer-based role-playing games. 2005. Citado na página 31.

MARCZEWSKI, A. *Gamification: a simple introduction*. [S.l.]: Andrzej Marczewski, 2012. Citado na página 33.

MORGAN, G. A.; HARMON, R. J.; MASLIN-COLE, C. A. Mastery motivation: Definition and measurement. *Early education and Development*, Taylor & Francis, v. 1, n. 5, p. 318–339, 1990. Citado na página 14.

NAVARRO, E. *SimSE: a software engineering simulation environment for software process education*. Tese (Doutorado) — California State University at Long Beach, Long Beach, CA, USA, 2006. AAI3243955. Citado na página 6.

NEELI, B. K. Gamification in the enterprise: Differences from consumer market, implications, and a method to manage them. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 489–511. Citado na página 33.

NIELSEN, J. Going the distance in academic libraries: Identifying trends and innovation in distance learning resources and services. *Journal of Library & Information Services in Distance Learning*, Taylor & Francis, v. 8, n. 1-2, p. 5–16, 2014. Citado na página 33.

OLIVEIRA, F. F.; SANTOS, S. M. A arte de fidelizar clientes como diferencial competitivo. *Revista Foco*, v. 7, n. 2, 2014. Citado na página 34.

PAGELS, M. et al. Assessing the viability of implicitly estimated velocity for measuring the productivity of software teams. 2013. Citado 3 vezes nas páginas 21, 22 e 23.

PEDREIRA, O. et al. Gamification in software engineering–a systematic mapping. *Information and Software Technology*, Elsevier, v. 57, p. 157–168, 2015. Citado na página 44.

RABIN, S. Introduction to game development (game development). Charles River Media, Inc., 2005. Citado 2 vezes nas páginas 13 e 14.

RITTERFELD, U.; CODY, M.; VORDERER, P. *Serious games: Mechanisms and effects*. [S.l.]: Routledge, 2009. Citado na página 35.

ROYCE, W. W. Managing the development of large software systems. In: LOS ANGELES. *proceedings of IEEE WESCON*. [S.l.], 1970. v. 26, n. 8. Citado na página 21.

RUGGIERO, D. Gamification: Learning innovation or potential pitfall? *INTED2013 Proceedings*, IATED, p. 5190–5192, 2013. Citado na página 35.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, Springer, v. 14, n. 2, p. 131–164, 2009. Citado 3 vezes nas páginas 37, 38 e 39.

RUNESON, P. et al. *Case study research in software engineering: Guidelines and examples*. [S.l.]: John Wiley & Sons, 2012. Citado 3 vezes nas páginas 37, 38 e 39.

SCHACHT, S.; MAEDCHE, A. Project knowledge management while simply playing! gaming mechanics in project knowledge management systems. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 593–614. Citado na página 33.

SCHELL, J. *The Art of Game Design: A book of lenses*. [S.l.]: CRC Press, 2008. Citado na página 28.

SCHWABER, K. *Agile Project Management With Scrum*. Redmond, WA, USA: Microsoft Press, 2004. ISBN 073561993X. Citado 3 vezes nas páginas 5, 17 e 18.

SILVA, D. G. da. *Gamification of Scrum*. 2015. <http://www.agilegamification.org/gamification-of-scrum>. Citado na página 45.

SIMÕES, J.; REDONDO, R. D.; VILAS, A. F. A social gamification framework for a k-6 learning platform. *Computers in Human Behavior*, Elsevier, v. 29, n. 2, p. 345–353, 2013. Citado na página 33.

SINGER, L.; SCHNEIDER, K. It was a bit of a race: Gamification of version control. In: IEEE. *Games and Software Engineering (GAS), 2012 2nd International Workshop on*. [S.l.], 2012. p. 5–8. Citado na página 43.

SNIPES, W. B. Evaluating developer responses to gamification of software development practices. 2013. Citado na página 43.

STOCKINGER, T. et al. Towards leveraging behavioral economics in mobile application design. In: *Gamification in Education and Business*. [S.l.]: Springer, 2015. p. 105–131. Citado 2 vezes nas páginas 33 e 34.

SWEEDYK, E.; KELLER, R. M. Fun and games: a new software engineering course. *SIGCSE Bull.*, ACM, New York, NY, USA, v. 37, p. 138–142, June 2005. ISSN 0097-8418. Disponível em: <http://doi.acm.org/10.1145/1151954.1067485>. Citado na página 6.

SWEETSER, P.; WYETH, P. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)*, ACM, v. 3, n. 3, p. 3–3, 2005. Citado 2 vezes nas páginas 26 e 29.

TAKAHASHI, D. *Gamification gets its own conference*. 2010. <http://venturebeat.com/2010/09/30/gamification-gets-its-own-conference/>. Citado 2 vezes nas páginas 6 e 33.

TYCHSEN, A. Role playing games: comparative analysis across two media platforms. In: MURDOCH UNIVERSITY. *Proceedings of the 3rd Australasian conference on Interactive entertainment*. [S.l.], 2006. p. 75–82. Citado 2 vezes nas páginas 30 e 31.

TYCHSEN, A. et al. The game master. In: *Proceedings of the Second Australasian Conference on Interactive Entertainment*. Sydney, Australia, Australia: Creativity & Cognition Studios Press, 2005. (IE '05), p. 215–222. ISBN 0-9751533-2-3. Disponível em: <http://dl.acm.org/citation.cfm?id=1109180.1109214>. Citado na página 30.

TYCHSEN, A. et al. Live action role-playing games control, communication, storytelling, and mmorpg similarities. *Games and Culture*, SAGE Publications, v. 1, n. 3, p. 252–275, 2006. Citado na página 31.

VERSIONONE. *State of Agile Development*. 2010. Http://www.versionone.com/state_of_agile_development_survey/10/default.asp [last accessed: 2011-04-04]. Disponível em: <http://www.versionone.com/state_of_agile_development_survey/10/default.asp>. Citado na página 5.

WOHLIN, C. et al. *Experimentation in software engineering*. [S.l.]: Springer, 2012. Citado 3 vezes nas páginas 37, 38 e 81.

YIN, R. K. *Case study research: Design and methods*. [S.l.]: Sage publications, 2014. Citado na página 37.

ZICHERMANN, G.; CUNNINGHAM, C. *Gamification by design: Implementing game mechanics in web and mobile apps*. [S.l.]: "O'Reilly Media, Inc.", 2011. Citado 3 vezes nas páginas 29, 33 e 34.

Parte V

Apêndices

APÊNDICE A – Roteiro de realização da entrevista

1. Informar da realização e natureza da pesquisa: estudo de caso realizado com a equipe de desenvolvedores da Infoway.
2. Apresentar informações gerais sobre a entrevista
   a) o áudio a entrevista está sendo gravadas para posterior análise;
   b) as informações coletadas só serão utilizadas para a pesquisa e a privacidade dos entrevistados será preservada.
3. Apresentar o contexto de produtividade definido na pesquisa
4. Questionar sobre a classificação de produtividade da Infoway, da equipe do entrevistado e da sua própria produtividade. Se o entrevistado for um líder de equipe, interrogar sobre a produtividade dos membros da equipe.
5. Questionar aspectos que o entrevistado acredita serem decisivos para as produtividades apresentadas na questão anterior.
6. Informar que outras "conversas"podem acontecer, formalmente ou informalmente durante o andamento da pesquisa.

APÊNDICE B – Questionário aos desenvolvedores

Este questionário faz parte de estudo de caso realizado pelo Danilo Medeiros e Pedro de Alcântara com foco no cotidiano de desenvolvimento de software da Infoway. Ao responder tente dar informações detalhadas. As informações aqui coletadas poderão ajudar na transformação e melhoria do seu ambiente e cotidiano de trabalho! :) Nos últimos meses, o *redmine* recebeu funcionalidades como pontuação de tarefas, pedras preciosas, profile, notificações. As questões a seguir dizem respeito a influência desses aspectos no seu cotidiano de desenvolvimento.

1. Como você avalia a influência das funcionalidades no SEU cotidiano de desenvolvimento ?

   [ ] Para ser sincero eu nem as notei.

   [ ] Teve alguma influência, mas muito pouca.

   [ ] Influenciou razoavelmente, mas somente por um período de tempo, depois não mais.

   [ ] Alterou bastante o meu modo de trabalho, mas não foi duradouro.

   [ ] Alterou bastante e ainda continua influenciando o meu modo de trabalho.

   [ ] Outro(Campo aberto)

2. Descreva as influências Positivas que tenha percebido?

3. Descreva as influências Negativas que tenha percebido?

4. Como você avalia a influência dessas funcionalidades no cotidiano de desenvolvimento da sua EQUIPE?

   [ ] Não houve qualquer influência.

   [ ] Teve alguma influência, alguns fora influenciados, outros não.

   [ ] Influenciou razoavelmente, mas somente por um período de tempo, depois não mais.

   [ ] Alterou bastante o nosso modo de trabalho, mas não foi duradouro.

   [ ] Alterou bastante e ainda continua influenciando nosso modo de trabalho.

   [ ] Outro(Campo aberto)

5. Descreva as influências Positivas em sua equipe?

6. Descreva as influências Negativas na sua equipe?

7. Quanto a continuidade das funcionalidades:

   [ ] Acho que deveria continuar, é bem legal isso no dia a dia.

   [ ] Acho que deveria continuar, mas com correções e ajustes.

   [ ] Acho legal que permaneça, mas com mecanismos completamente diferentes.

   [ ] Tanto faz, pois de nada muda o nosso modo de trabalho.

   [ ] Deveria ser removida, pois mais atrapalha do que ajuda.

   [ ] Outro(Campo aberto)

8. Assumindo que produtividade seja a sua capacidade ou da sua equipe de concluir tarefas, considere uma escala de 0 a 10 responder as seguintes questões: Escala: 0: Conclui a iteração sem conseguir concluir nenhuma tarefa a contento; 5: Produtividade ruim, entrega parcialmente com dificuldades; 7: Produtividade aceitável, mas com grandes pontos de melhoria; 10: Produtividade mais que satisfatória: eficaz e eficiente.

9. Como você classificaria a sua atual produtividade?

10. E a produtividade da sua equipe?

11. E a produtividade da Infoway?

12. Você acha que a sua produtividade, da sua equipe ou da Infoway são influenciadas de alguma forma pelas funcionalidades adicionadas no redmine? Como? Comente também as classificações que você definiu.