UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RENATA LOPEZ MARQUES

UMA ABORDAGEM PARA TESTES DE SOFTWARE NA ÁREA DE *GAMES*

NITERÓI
2009

RENATA LOPEZ MARQUES

UMA ABORDAGEM PARA TESTES DE SOFTWARE NA ÁREA DE *GAMES*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Áreas de Concentração: Engenharia de Software, Desenvolvimento de *Games*

Orientador: Prof. Dr. LEONARDO GRESTA PAULINO MURTA

Niterói

2009

RENATA LOPEZ MARQUES

UMA ABORDAGEM PARA TESTES DE SOFTWARE NA ÁREA DE *GAMES*

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Áreas de Concentração: Engenharia de Software, Desenvolvimento de *Games*

Aprovada em dezembro de 2009.

BANCA EXAMINADORA

Prof. Dr. LEONARDO GRESTA PAULINO MURTA - Orientador

UFF

Prof. Dr. ESTEBAN WALTER GONZALEZ CLUA

UFF

Prof. Dra. TERESA CRISTINA DE AGUIAR

UFF

NITERÓI

2009

Aos meus pais, Sandra e Antônio, minha irmã Fernanda, meus avós Ascensión, Belarmino, Lúcia e Antônio, meu namorado Diego e a todos que me ajudaram e me incentivaram ao longo deste trabalho.

AGRADECIMENTOS

A Deus, à Universidade Federal Fluminense, ao Professor Doutor Leonardo Gresta Paulino Murta pela dedicação na orientação deste trabalho, à minha família pelo apoio, ao meu namorado Diego pela companhia e compreensão, a todos os professores, funcionários e amigos que tive o prazer de conhecer e conviver durante o tempo de graduação, em especial aos amigos Gustavo, Adrian, Rafael Moulin, Andre, Willen e à empresa Aiyra.

RESUMO

Lopez Marques, Renata. Uma Abordagem para Testes de Software na Área de *Games*. Niterói, 2009. 63 p. Trabalho de Conclusão de Curso – Instituto de Computação, Universidade Federal Fluminense.

Consumidores estão sempre buscando qualidade em produtos ou serviços que compram. É essencial que um software supra as necessidades de um usuário, e é papel da equipe de desenvolvedores fazer com que, no final, o cliente fique satisfeito. Um software que não funciona ou que apresente defeitos não atende a esse requisito.

Essa história não é diferente com um tipo de software em particular: Jogos Eletrônicos, ou *Games*. Essa indústria cresce assustadoramente com o passar do tempo no mundo todo, rendendo bilhões por ano. Com novas técnicas e softwares para desenvolvimento surgindo de tempos em tempos, empresas de *games* se tornam cada vez mais profissionais e especializadas em um produto de qualidade. Além disso, consumidores se tornam sempre mais exigentes, buscando um produto que apresente diferencial e inovação com relação aos lançados anteriormente. Produtos novos com qualidade inferior a produtos mais antigos são descartados. Portanto, para disputar com empresas fortes, é necessário, no mínimo, garantir um produto de qualidade.

Para tentar minimizar o número de falhas em um software, entra em cena a equipe de Testes de Software, cujo papel é testar o sistema em toda fase de desenvolvimento, buscando erros e inconsistências que prejudicam a funcionalidade do sistema. Essa equipe é importante porque garante uma melhor qualidade do software antes da entrega e evita maiores prejuízos em supostas correções aos defeitos. Além disso, mantém uma boa imagem da empresa e da equipe de desenvolvimento.

Em se falando da indústria de *games*, um tipo específico de equipe de testes atinge um alto nível de importância: Os *Game Testers*, pessoas cujo trabalho consiste em jogar o g*ame* a ser lançado e descobrir falhas de integração entre as áreas do desenvolvimento que põem em risco a integridade do produto.

Esse trabalho mostra a importância dos testes na área de g*ames*, abordando uma nova estratégia com tarefas de testes realizadas pela equipe de desenvolvimento com a perspectiva de *Game Testers,* para ser aplicada na etapa de desenvolvimento de um g*ame* e minimizar os problemas que esta área oferece aos testes.

Palavras-chave

Desenvolvimento de *Games*, Metodologia de Desenvolvimento de Software, Engenharia de Software, Testes de Software, Metodologia Ágil, *Scrum*, adaptação, *Game Tester*, *Quality Assurance*

ABSTRACT

Lopez Marques, Renata. An Approach to Software Testing for Games. Niterói, 2009. 63 p. Term Paper – Instituto de Computação, Universidade Federal Fluminense.

Consumers are always looking for products or services with quality. It is essential that a software meets the needs of a user, and it's the development team's responsibility to keep the customer satisfied. A software that does not work or that is defective does not meet this requirement.

This story is not different with a type of software in particular: Games. This industry is growing dramatically over time, yielding billions a year. With new software development techniques arising from time to time, game companies are becoming more professional and specialized in quality products. Moreover, consumers become ever more demanding, seeking a product possessing differential and innovation. New products with lower quality compared to older products are discarded. Therefore, to compete with strong companies, you must at least ensure a product with quality.

To try to minimize the number of faults in software, we have the Software Testing team, whose role is to test the system at every stage of development, seeking to errors and inconsistencies that undermine the system's functionality. This team is important because it ensures a better quality for the software prior to delivery and prevents further damage to the supposed fixes to defects. Additionally, it has a good corporate image and team development.

In speaking of the games industry, a specific type of test team achieves a high level of importance: The Game Testers, people whose job is to play the game to be released and find failures of integration between the areas of development which challenge risk the integrity of the product.

This study aims to show the importance of tests in the games, covering a new strategy with tests made by the development team with *Game Designers*' perspective, to implement the development stage of a game and minimize the problems between game development and tests.

Keywords

Game Development, Software Development Methodology, Software Engineering, Software Testing, Agile Methodology, Scrum, adaptation, Game Tester, Quality Assurance, software development methodologies

SUMÁRIO

Lista de Figuras

1. Introdução

1.1 Motivação

Segundo dados colhidos pelo NPD Group (PC WORLD, 2009) e divulgados pela *Entertainment Software Association* (ESA) em 25 de janeiro de 2008, as vendas da indústria de *games* atingiram receita de 9,5 bilhões de dólares em 2007. Isso significa que essa indústria ultrapassa os números das indústrias áudio-visuais (FELICIANO, 2007), até pouco tempo consideradas as maiores indústrias de entretenimento do planeta.

Os jogos eletrônicos, ou *Games*, estão atingindo um alto grau de importância na economia global. É fácil notar as repercussões quando algum *game* esperado é lançado, ou quando surgem rumores de novos desenvolvimentos por empresas conhecidas no ramo.

Não deixa de ser estranho notarmos que os *games* vêm agindo na sociedade com cada vez mais influência. São organizados diversos eventos por ano, tanto para desenvolvedores quanto para os jogadores. Depoimentos e histórias de pessoas que de fato desenvolvem um vício por jogos eletrônicos e precisam de reabilitação vêm se tornando cada vez mais comuns (REGIS, 2009). Pesquisadores suecos (FOLHA ONLINE, 2010) chegaram a afirmar que *World of Warcraft* (BLIZZARD, 2010) – hoje um dos *games* mais populares do mundo, com mais de 11 milhões de usuários – pode ser tão viciante quanto crack. A indústria de Informática e Hardware, assim como indústrias de Tecnologia em geral, faturam com os usuários que buscam cada vez mais equipamentos com mais recursos, para atenderem às restrições funcionais dos novos *games* desenvolvidos.

Com isso, é possível vermos que essa massa de jogadores tende a ser cada vez mais exigente quanto à qualidade do produto a ser vendido. *Games* inferiores em quaisquer aspectos são fadados ao fracasso de aceitação em um mercado tão vasto como este. Por isso, a qualidade dos *games* deve ser garantida, de forma a corresponder às expectativas do público alvo, mantendo assim a confiabilidade e a boa imagem da equipe de profissionais responsável pelo desenvolvimento do *game*. A tarefa de testes se apresenta como aliada na busca dessa garantia.

1.2 Objetivo

Esta monografia tem como objetivo apresentar técnicas de testes no desenvolvimento de *games* feitos com o uso de uma metodologia ágil, abordando uma estratégia de testes onde os próprios desenvolvedores[1] atuem como *Game Testers*, bem como analisar os desafios que esta área apresenta aos testes de software. Ao longo do trabalho, esses desafios são discutidos, com o objetivo de formular uma abordagem de testes que aumente a qualidade do produto.

O cenário de estudo para este trabalho é um *game* em desenvolvimento por uma equipe de profissionais na empresa Aiyra, incubada pela Universidade Federal Fluminense.

1.3 Estrutura do trabalho

Este trabalho está organizado de forma a ambientar o leitor com os temas propostos, fornecendo inicialmente uma revisão dos dois principais assuntos - Testes de Software e *Games* - e, posteriormente, analisando a integração dessas duas áreas de forma a desenvolver uma abordagem de testes.

O capítulo 2 apresenta uma breve revisão sobre Testes de Software, enfocando a divisão de tarefas, diferenciação de etapas e abordando a importância dos testes atualmente.

O capítulo 3 apresenta conceitos gerais da área de *games* e discute o desenvolvimento de *games* sob uma ótica ágil, relacionando as expectativas com a qualidade do produto final, e o que é preciso para que a última corresponda a essas expectativas.

O capítulo 4 apresenta uma abordagem de testes integrada ao desenvolvimento de *games*, aplicada a um *game* em desenvolvimento pela empresa Aiyra, de forma a melhorar sua qualidade e minimizar os problemas que essa integração apresenta.

O capítulo 5 apresenta a conclusão geral deste trabalho, de forma a resumir as idéias principais discutidas e apresentando as contribuições desta monografia para o ramo citado, bem como as limitações encontradas e possíveis trabalhos futuros.

---

[1] Por *desenvolvedores*, entende-se todos os membros das equipes de produção de um *game*.

2. Testes de Software

2.1 Introdução

Há algum tempo, programadores e desenvolvedores de softwares em geral trabalhavam atentamente, visando a integridade do código e, assim, uma boa qualidade do software. Testar o código era visto como uma simples revisão e considerada uma tarefa tediosa e pouco importante, muitas vezes sendo até evitada (NOGUEIRA, 2009).

Hoje, a boa qualidade de um sistema vem sendo cada vez mais exigida. Um software deve desempenhar seu papel de forma rápida e sucinta. Se houver erros, a imagem dos desenvolvedores é prejudicada.

A necessidade de um software de qualidade cresceu (e vem crescendo) assustadoramente, já que praticamente tudo é controlado ou aperfeiçoado através de software. A qualidade já se tornou um requisito básico. Erros significam prejuízos, por isso, só o cuidado no planejamento e na programação não são mais o bastante para garantir a qualidade de um sistema.

É nesse contexto que entra o conceito formal de Testes de Software. As estratégias de testes de software ajudam a detectar erros decorrentes de defeitos que foram introduzidos no momento em que o software foi projetado e construído, mas passaram desapercebidos. Dessa forma, os testes não demonstram o bom funcionamento de um programa, mas sim os erros nele contidos.

Para uma melhor compreensão da terminologia básica sobre qualidade de software, devemos ter em mente algumas definições: **Erros** (do inglês, *mistakes*) são ações cometidas por pessoas (desenvolvedores). Esses erros, quando introduzidos no software, geram **defeitos**. Esses defeitos, quando (e se) executados, resultarão em uma **falha** de processamento, que gera um erro (do inglês, *error*). Por isso, quando falamos de qualidade de um software, estamos nos referindo apenas a defeitos e falhas (BERGAMI, 2008).

O teste de software é freqüentemente referido como Verificação e Validação (PRESSMAN, 2006). A diferença entre elas é sutil: Enquanto a verificação se refere a

atividades que testam se o software funciona livre de erros, a validação se refere a atividades que testam se o software funciona de acordo com os requisitos do cliente, ou seja, se faz o que foi pedido. BOEHM (1979) propõe a análise dessa diferença através de um ponto de vista interessante:

- Verificação: Estamos construindo o produto corretamente?
- Validação: Estamos construindo o produto certo?

De uma maneira menos formal, testes avisam se o programador escreveu o código que de fato queria escrever (PILONE e MILES, 2007).

Testar significa pensar em possíveis erros, e tratá-los. Significa evitar perda de tempo e dinheiro. A grande maioria dos erros que ocorreriam após a entrega de um sistema deveria ser detectada e corrigida no processo de testes.

Não é certo, contudo, assumir que os testes eliminam por completo quaisquer erros em um sistema. Por mais que todas as partes de um código sejam submetidas a diversos testes, não podemos saber se algum defeito surgirá depois do software entregue. O que pode ser feito é imaginar quais erros poderiam vir a se manifestar, e tentar tratá-los.

Estudos mostram que a maior quantidade de erros são cometidos na fase inicial do projeto de desenvolvimento, onde não há uma boa especificação de requisitos (TOZELLI, 2008). Com isso, defeitos vão se propagando ao longo das etapas de desenvolvimento e aumentando proporcionalmente os custos de correção. Por isso, é essencial que os testes comecem logo no início do projeto, pois diminuirão os defeitos que se propagam para fases mais avançadas, aumentando exponencialmente o problema. A figura 1 acompanha o custo de um defeito ao longo do desenvolvimento.

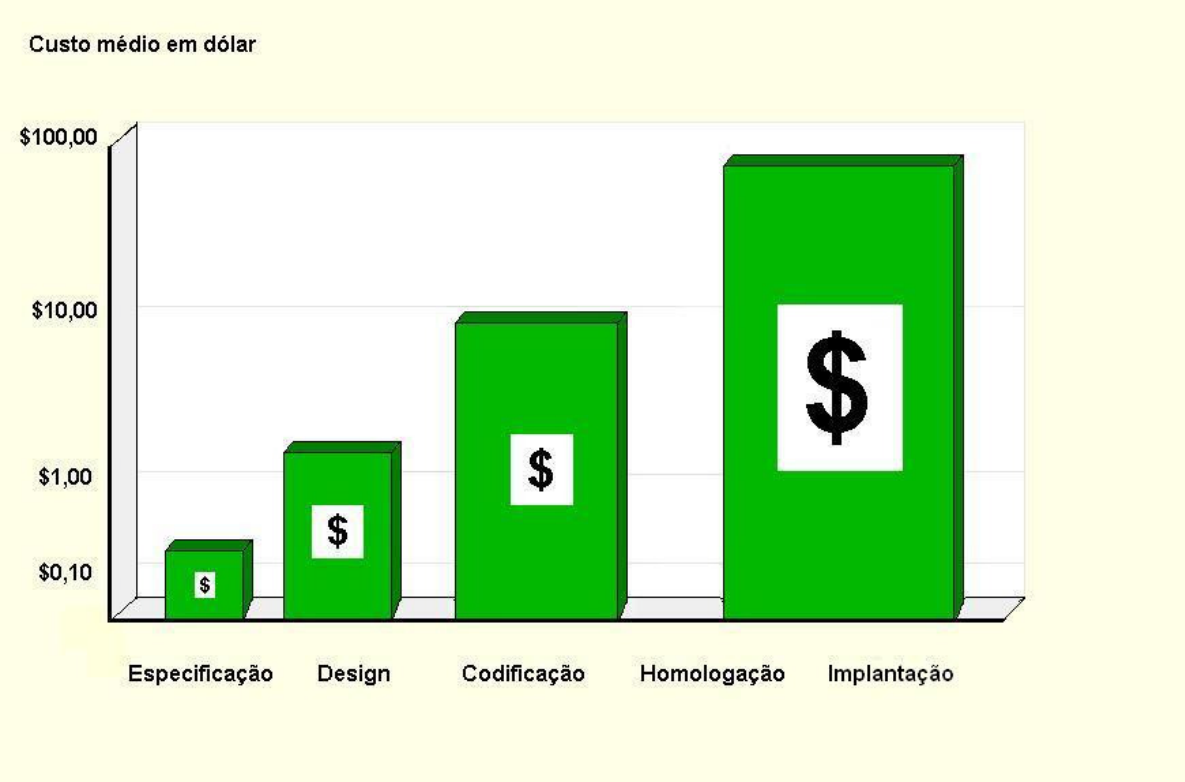

Figura 1 – Os custos de um defeito crescem em uma proporção média de dez vezes a cada etapa de desenvolvimento (PALMA, 2007).

Ainda assim, não podemos ter certeza que não há propagação de defeitos. Por isso, conforme o projeto vai crescendo, sua complexidade vai aumentando. Cabe a uma série de profissionais administrar e modelar testes, da forma mais abrangente e menos complexa possível.

Contudo, para garantir o sucesso de uma empresa de desenvolvimento de software, não basta que a equipe de desenvolvedores teste o sistema e o entregue sem defeitos. O processo de teste gera métricas e números. Sem a análise dos resultados obtidos, não há como saber se, de fato, houve benefício para o projeto. Além disso, decisões e mudanças importantes para o projeto podem não ser tomadas por falta de informações exatas, que deveriam ser coletadas nos resultados dos testes. Estratégias e objetivos devem ser traçados no início do processo, para que haja uma base para a análise de resultados.

Nas próximas seções, serão mais detalhados os papéis na área de Testes de Software, bem como os diferentes níveis desta área, os diferentes pontos de vista para a execução de testes e as etapas do processo.

2.2 Papéis na Área de Testes de Software

O processo de Testes de Software tem uma série de profissionais que possuem funções bem definidas dentro da equipe. Segundo BARTIE (2006), alguns dos profissionais dessa área são os seguintes:

▪ Líder de Testes

É o responsável pela qualidade do projeto de testes, certificando-se que todas as atividades de testes estejam sendo realizadas nos prazos, custos e qualidade contratados pelo Cliente. Participa ativamente dos trabalhos e avalia a qualidade dos resultados, podendo interferir nas atividades executadas pelos profissionais de Testes. O Líder de Testes garante o sucesso do projeto como um todo. Por isso, é uma figura muito presente em todas as etapas do processo. Revisa os resultados e faz comparações com trabalhos anteriores, aponta falhas da documentação e avalia a organização das atividades.

▪ Analista de Testes

Responsável por avaliar e garantir a qualidade de cada aplicativo isoladamente e garantir que os requisitos de negócios estão sendo adequadamente suportados pelo sistema. Seu papel é ampliar a cobertura[2] dos casos de testes[3], identificando novas formas e possibilidades de aumentar a cobertura dos testes. Deve buscar sempre aprimorar os níveis de qualidade, prazos e custos já alcançados nos projetos anteriores, buscando eliminar retrabalhos e simplificar o processo de construção de novos casos de testes. Além disso, executa os testes progressivos[4] e os testes regressivos[5], conferindo os resultados.

▪ Executor de Testes

Apesar de sua responsabilidade ser mais focalizada na etapa de Execução de Testes, sua presença nas etapas anteriores é fundamental, pois facilita o entendimento do trabalho que realizará nas próximas etapas. Este profissional deve garantir a qualidade dos aplicativos, executando testes progressivos e regressivos (além de conferir seus resultados), dando apoio

---

[2] Medida da abrangência do teste.

[3] Conjunto de condições usadas para o teste que especificam o resultado esperado (para comparação com o resultado obtido).

[4] Teste das funcionalidades especificadas para a nova versão.

[5] Testes das funcionalidades que não foram afetadas pela nova versão (que não deveriam ter sido afetadas).

na identificação dos casos de testes e detalhando as massas de entrada e saída aplicadas a cada caso de teste.

▪ Gerente de Testes

Responsável pelo gerenciamento dos Serviços de Testes de Software, controlando e monitorando continuamente o desempenho dos projetos de testes. Deve garantir que o nível de serviço de testes ocorra dentro das expectativas de prazo, custo e qualidade que o cliente definiu, acompanhando o andamento e avaliando o desempenho dos Projetos de Testes, além de realizar o planejamento financeiro de todos os Projetos.

▪ Auditor de Testes

Profissional responsável por identificar desvios nos procedimentos previstos e estabelecer planos de ação, evitando perda de padronização e descontrole do processo. Para isso, o Auditor de Testes mede e avalia continuamente o Processo de Testes, avalia desvios de prazo, custo e qualidade dos projetos, identifica e corrige não-conformidades dos projetos e avalia o nível de satisfação dos clientes (*feedback*).

▪ Arquiteto de Testes

Responsável pela evolução dos indicadores de produtividade e qualidade dos projetos de testes de software, atua diretamente com equipes de testes e planeja a construção de arquiteturas que suportam diferentes ambientes de testes. Monta a infra-estrutura de testes, aumentando a velocidade e a precisão dos testes, analisando as restrições tecnológicas, gerenciando ferramentas e padronizando as soluções de testes.

▪ Automatizador de Testes

Responsável pela implementação de scripts de testes (automações) e construção de simuladores que compõem a arquitetura de testes de um aplicativo, visa a disponibilidade e confiabilidade das arquiteturas de testes e está sempre incrementando funcionalidades para melhorar a produtividade da equipe de testes. Trabalha junto com o Arquiteto e com o Analista de Testes, para melhorar a compreensão dos diferentes aplicativos.

2.3 Níveis de Testes de Software

Segundo NAIK e TRIPATHY (2006), o processo de testes deve atuar em diferentes níveis, envolvendo partes do sistema. Os quatro níveis são Unidade, Integração, Sistema e Aceitação. Enquanto nos três primeiros níveis os testes são conduzidos pela equipe de desenvolvimento, o último é realizado pelo próprio cliente.

AMMANN e OFFUTT (2008) relacionam os diferentes níveis de testes com suas respectivas atividades de desenvolvimento de software:

- Teste de Unidade – avalia o software em relação à implementação.
- Teste de Integração – avalia o software em relação ao *design* do subsistema.
- Teste de Sistema – avalia o software em relação ao *design* da arquitetura.
- Teste de Aceitação – avalia o software em relação aos requisitos.

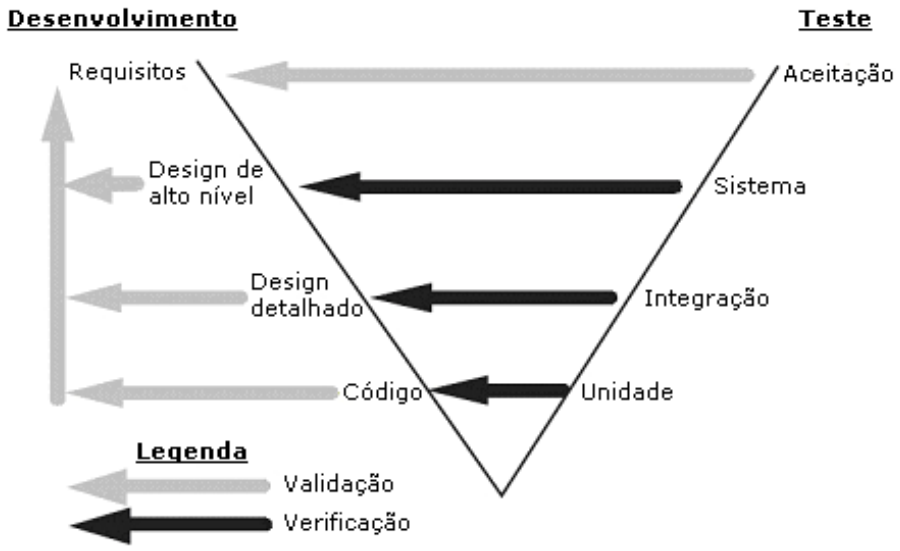

Figura 2 – Fases de teste e desenvolvimento de um software (adaptado de NAIK e TRIPATHY, 2006).

Nos **Testes de Unidade**, são testadas individualmente as unidades do programa, como procedimento, funções, métodos ou classes, de forma a encontrar falhas de funcionamento no código, dentro de uma pequena parte do sistema independentemente do todo. Nas palavras de

PRESSMAN (2006), “o Teste de Unidade enfoca a lógica interna de processamento e as estruturas de dados dentro dos limites de um componente”.

Após os módulos terem sido testados no Teste de Unidade, eles devem ser colocados em conjunto, ou seja, os componentes internos do sistema devem ser integrados. Aí entram os **Testes de Integração**, que são executados com a finalidade de encontrar erros de comunicação entre as interfaces dos módulos. Um tipo de teste muito importante no Teste de Integração é o Teste de Regressão, que consiste em re-executar testes que já foram executados para garantir que modificações provenientes da adição de um novo módulo não propaguem erros.

Testes de Sistema determinam se o sistema funciona como um todo. Segundo PRESSMAN (2006), os Testes de Sistema incluem testes de recuperação (verificando se a recuperação do sistema a falhas é realizada de forma adequada), testes de segurança (verificando se os mecanismos de proteção de um sistema o protege de invasões), testes de stress (analisando como o sistema reage à demanda de recursos em quantidade, freqüência e volume anormais) e testes de desempenho (testando o desempenho do software durante a execução).

Testes de Aceitação são realizados geralmente pelos próprios usuários finais e são formulados para verificar se o software de fato consegue suprir as necessidades dos clientes, ou seja, se faz o que foi pedido. Quando feitos por testadores, mesmo que esses testes aprovem o software, ainda há chances de se entregar ao cliente o produto com erros, já que pode ter havido, no início do projeto, uma má interpretação dos requisitos dados pelo usuário.

2.4 Testes através de perspectivas

Segundo PILONE e MILES (2007), há três pontos de vista ao se observar um sistema, que deve ser testado através de todas essas perspectivas:

2.4.1 Ponto de vista do usuário

Nesta técnica de teste, também chamada de Teste Funcional, o usuário vê o sistema de fora, ou seja, não se importa com o código ou com qualquer comportamento interno, só se interessando com a funcionalidade do sistema: ou o sistema faz o que o usuário pediu, ou não faz.

Como as características internas do sistema passam "escondidas" aos olhos dos testadores, essa técnica é comumente conhecida como Caixa-Preta.

Dessa forma, o teste é executado partindo de dados de entrada e o resultado obtido é comparado com o resultado esperado pelo que foi entendido dos requisitos do cliente. (Se for o próprio cliente a realizar esse teste, talvez aí sejam encontrados alguns erros nos requisitos, já que o software pode não estar fazendo o que o usuário desejava).

2.4.2 Ponto de vista do testador

O Testador se preocupa, assim como o usuário, com a funcionalidade do sistema, verificando se ele se comporta da forma que deveria se comportar. Porém, através deste ponto de vista, o testador busca também saber se o código está rodando de uma maneira correta para que essas funcionalidades estejam certas.

Assim, alguém que testa o sistema desse ponto de vista está procurando basicamente pelas mesmas coisas que um testador do ponto de vista do usuário. A diferença é que, neste caso, é possível que o testador olhe também por baixo da superfície do software e verifique a consistência do código.

Por estabelecer um "meio termo" entre os pontos de vista do testador e do desenvolvedor, essa técnica de teste também é conhecida como Caixa Cinza

2.4.3 Ponto de vista do desenvolvedor

Nesta técnica de teste, também chamada de Teste Estrutural, o testador não se preocupa se o sistema apresenta incompatibilidades com os requisitos dos usuários. Está

interessado apenas na integridade do código. O testador analisa os algoritmos do software, aplicando testes de forma a encontrar erros de compilação e execução.

Nesse ponto de vista, o testador precisa conhecer e entender o código e características internas específicas do sistema. Por isso, essa técnica de teste é também chamada de Caixa-Branca.

2.5 Etapas do processo de Teste de Software

O Processo de Teste de Software é dividido em etapas que ajudam a organizar e melhoram os resultados do processo. A essas etapas são dadas funções, que são coordenadas e executadas por diferentes profissionais de testes.

Segundo BARTIE (2007), essas etapas são as seguintes:

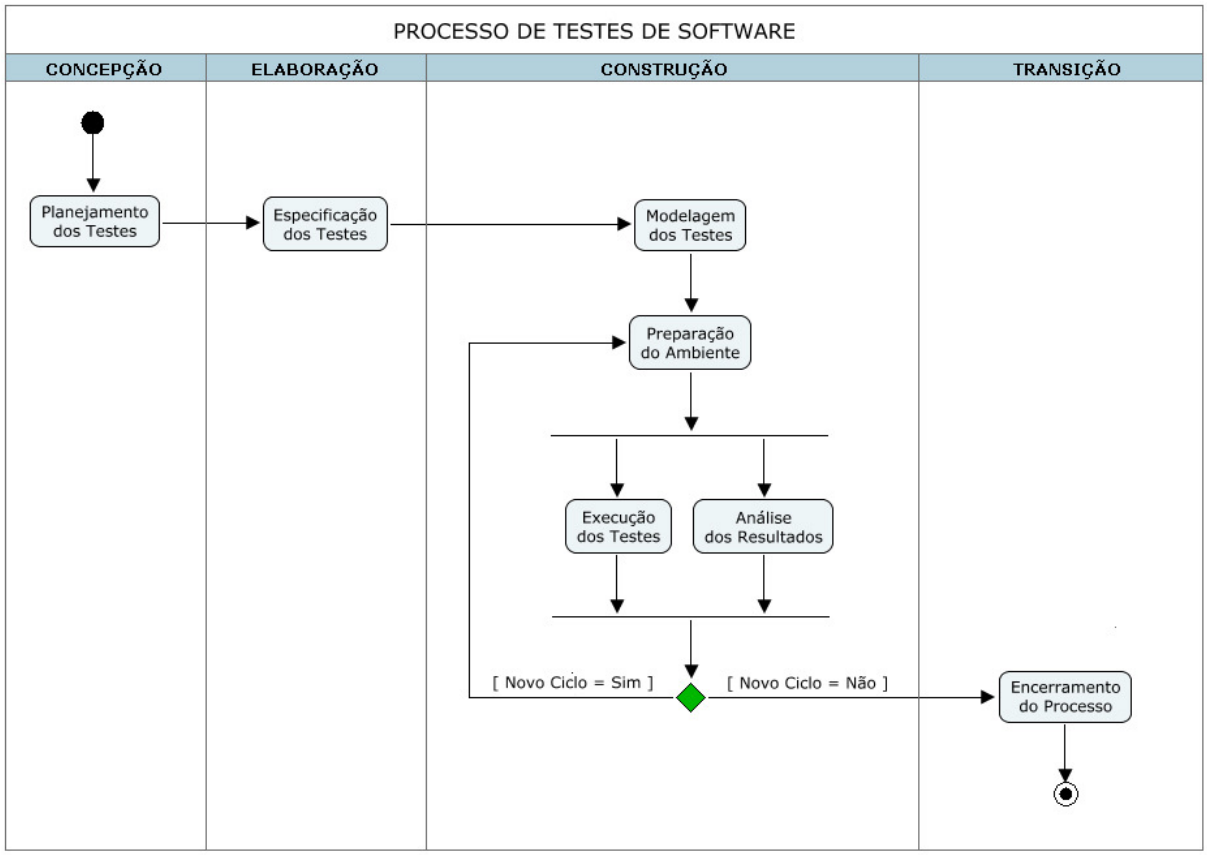

Figura 3 – Processo de Testes de Software (adaptado de BARTIE, 2007).

2.5.1 Planejamento de Teste

Caracteriza-se pela definição de uma proposta de testes baseada nas expectativas do cliente em relação a prazos, custos e qualidade, possibilitando dimensionar a equipe e estabelecer um esforço de acordo com as necessidades apontadas pelo cliente.

Dentro da etapa de Planejamento de Teste, são executadas as seguintes macro-atividades:

- Estudo do Projeto, que engloba estudos de novos requisitos por parte do cliente (modificações), estudos de modificação de arquiteturas e estudos de risco.

- Avaliação de Impacto, que engloba estudos sobre necessidade de adequações na automação dos testes e nas ferramentas empregadas, bem como a construção de novas ferramentas e modificações no ambiente.

- Definição de Cenários Possíveis, onde há avaliação de disponibilidade de recursos internos a serem alocados para o projeto, definição de riscos e planos de ação.

2.5.2 Projeto de Casos de Teste ou Especificação dos Testes

Esta etapa é caracterizada pela identificação dos casos de testes que deverão ser construídos e modificados em função das mudanças solicitadas pelo cliente, bem como pelo próprio aperfeiçoamento do processo de testes.

Dentro da etapa de Especificação dos Testes, são executadas as seguintes macro-atividades:

- Estudo dos requisitos, que engloba estudos de requisitos (funcionais e não funcionais) e mudança de requisitos por parte do cliente, identificação de inconsistências nos requisitos e obtenção de *feedback*.

- Especificação das adaptações da Arquitetura dos Testes[6], onde há especificação de adequação das ferramentas empregadas e de novas ferramentas exigidas, bem como especificação das modificações na organização do ambiente.

---

[6] Composição de elementos de automatização de testes.

• Identificação dos Casos de Testes, propondo as identificações de mudanças solicitadas pelo cliente e dos casos de uso de cada solicitação.

• Refinamento dos Casos de Testes, onde cada Analista de Testes apresenta seus próprios casos de teste enquanto um grupo de Analistas de Testes criticam esses casos e avaliam o nível de cobertura alcançado.

• Aceitação dos Casos de Testes, onde são identificadas áreas-chave para a apresentação dos casos de testes e refinados os casos de testes apresentados.

• Refinamento do Projeto de Testes, que propõe a reavaliação de riscos do projeto após um maior detalhamento dos requisitos e a negociação em relação às possíveis modificações no Projeto de Testes (como prazos e custos).

2.5.3 Modelagem de Teste

Esta etapa é caracterizada pela identificação de todos os elementos necessários para a implementação de cada caso de testes especificado. Fazem parte desta etapa a modelagem das massas de testes e a definição dos critérios de tratamento de arquivos.

Dentro da etapa de Modelagem de Testes, são executadas as seguintes macro-atividades:

• Criação dos Roteiros de Testes, onde roteiros de testes que atenderão a novos casos de testes são identificados e revisados e procedimentos para iniciar, executar e validar os casos de testes são especificados.

• Detalhamento da Massa de Entrada e Saída, onde novos pontos de simulação e validação são identificados e a massa de entrada e de saída de dados é detalhada (para cada caso de teste).

• Identificação de Critérios de Tratamento da Massa de Testes, onde são identificados os critérios de descaracterização[7], envelhecimento[8] e redução[9] da massa de testes.

• Implementação das Adaptações da Arquitetura dos Testes, que engloba a implementação da adequação nas ferramentas usadas e das ferramentas exigidas, implementação das modificações estruturais no ambiente e implementação das adequações na automação da preparação do ambiente, execução de testes e análise dos resultados.

• Elaboração do Plano de Execução dos Testes, que engloba a identificação das estações de trabalho e dos equipamentos que serão necessários para a execução dos testes, bem como as configurações que serão exigidas, identificação das licenças de softwares que serão utilizados e agrupamento dos casos de testes por características.

2.5.4 Preparação do Ambiente

Esta etapa é caracterizada por um conjunto de atividades que visa a disponibilização física de um ambiente de testes "segregado" que esteja pronto para sofrer a bateria de testes planejadas nas etapas anteriores.

Dentro da etapa de Preparação do Ambiente, são executadas as seguintes macro-atividades:

• Instalação do Aplicativo a ser Testado, onde são identificadas e baixadas a versão do aplicativo (e de seus componentes) que será testado e a versão do Banco de Dados.

• Instalação da Arquitetura de Testes, onde é testada a compatibilidade entre a versão da arquitetura de testes e a versão do aplicativo a ser testado, e baixadas as versões do Banco de Dados, dos simuladores e componentes da arquitetura, dos scripts e de ferramentas para o processo de teste.

---

[7] Alterações nos dados (mascarar dados), de forma a manter os dados sigilosos e a confidencialidade da empresa.

[8] Povoamento de dados envelhecidos, para atender a necessidades específicas dos testes.

[9] Diminuição da base de dados oferecida para testes.

• Homologação da Nova Arquitetura, que engloba a seleção de um conjunto de testes a fim de testar a arquitetura, a geração de Massa de Testes (entrada e saída) para a simulação e a avaliação dos resultados obtidos.

• Geração da Massa de Testes, onde são analisadas a versão da massa de testes e a versão do aplicativo testado, a fim de obter compatibilidade, geradas as massas de entrada e saída nos respectivos diretórios e avaliados os tratamentos das informações.

2.5.5 Execução de Teste

Esta etapa é caracterizada pela execução dos testes planejados, visando garantir que o comportamento do aplicativo atende aos requisitos dados pelo cliente.

Dentro da etapa de Execução de Teste, são executadas as seguintes macro-atividades:

• Execução de Testes Progressivos, onde, após esses testes serem feitos, evidências coletadas com resultados esperados e não-esperados são analisadas e casos de testes em conformidade e em suspeita de não-conformidade com os requisitos são identificados.

• Confirmação dos Resultados Progressivos, onde os casos de testes progressivos em suspeita de não-conformidade com os requisitos são re-executados, falsos positivos e duplos positivos são identificados e é apontada a imprecisão no processo de comparação (no primeiro caso) ou atestada a não-conformidade com os requisitos (no segundo caso).

• Execução de Testes Regressivos, onde, após esses testes serem feitos, evidências coletadas com resultados esperados e não-esperados são analisadas e casos de testes em conformidade e em suspeita de não-conformidade com os requisitos são identificados.

• Confirmação dos Resultados Regressivos, onde os casos de testes regressivos em suspeita de não-conformidade com os requisitos são re-executados, falsos positivos e duplos positivos são identificados e é apontada a imprecisão no processo de comparação (no primeiro caso) ou atestada a não-conformidade com os requisitos (no segundo caso).

2.5.6 Análise dos Resultados

Esta etapa é caracterizada pela avaliação e confirmação dos resultados relatados durante a fase de execução dos testes. Os resultados em "não-conformidade" deverão ser "confirmados" e "detalhados" para que a Fábrica de Software realize as correções necessárias. Já os casos de testes progressivos em "conformidade" deverão ter seu resultado "POSITIVO" reconfirmado e seu "*baseline*" atualizado.

Dentro da etapa de Avaliação de Resultados, são executadas as seguintes macro-atividades:

- Revisão dos Resultados em Não-Conformidade, onde casos de testes em conformidade e não-conformidade são identificados e falsos positivos e falsos negativos são apontados. A conformidade dos casos de testes é atestada (no primeiro caso) e imprecisões no processo de comparação e análise são apontadas (no segundo caso).

- Atualização do *Baseline*, onde casos de testes em conformidade e não-conformidade são identificados e o *baseline* é atualizado.

- Formalização dos Defeitos Detectados, onde cada caso de testes resultante em não-conformidade é isolado, as evidências que provam a não-conformidade são coletadas e os defeitos detectados são formalizados.

2.6 Considerações Finais

Vimos a importância dos Testes de Software para garantir a qualidade de um produto final, observando as diversas etapas do processo de testes e os papéis desempenhados em cada uma dessas etapas pelos membros de uma equipe de testes.

No próximo capítulo, serão apresentados conceitos gerais de *Games*. Também será discutido o desenvolvimento de *games* através de uma metodologia ágil e a importância da garantia de qualidade nesta área.

3. Games

3.1 Introdução

"*Games* são softwares com arte, áudio e jogabilidade" (BETHKE, 2003).

Desenvolvedores de *games* devem entender que, assim como no desenvolvimento de um software, a equipe deve seguir métodos de produção rigorosos e reservar tempo significativo na produção, de forma a atingir o produto esperado. Qualquer *game* deve ter uma metodologia de desenvolvimento. Sem ela, um *game* demora muito mais tempo para ser desenvolvido do que de fato deveria demorar, além de apresentar mais falhas e imperfeições (BETHKE, 2003).

Além disso, a equipe de desenvolvimento de um *game* (assim como com qualquer software) deve estar ciente de suas tarefas e se informar sobre as tarefas dos outros membros da equipe. A troca de informação dentro da equipe faz com que o trabalho se torne muito mais rápido e flua sem tantos problemas.

Desenvolvedores de *games* para computadores devem estar atentos aos diferentes tipos de máquinas e suas restrições de hardware e software. Para atingir o sucesso, um *game* deve funcionar de forma satisfatória na maioria das máquinas mais usadas. A grande variedade de sistemas operacionais e de componentes de hardware dificulta o trabalho das equipes de desenvolvimento, pois um *game* pode rodar de maneiras diferentes em computadores diferentes. É por isso que a equipe de desenvolvimento deve tentar ao máximo diminuir os requisitos mínimos necessários para uma boa jogabilidade, sem afetar na qualidade do produto. Contudo, desenvolvedores de *games* para consoles de videogames têm a vantagem de conhecerem suas plataformas, que são imutáveis (BETHKE, 2003).

Nas próximas seções, serão vistas as partes da produção de um *game.* Também serão discutidas a qualidade de *games* e a relação desta qualidade com o desenvolvimento de *games* através de uma metodologia ágil.

3.2 Partes da produção de Games

Geralmente, a produção de *games* é dividida em quatro partes: *Game Design*, Código, Arte e Som. É importante que o trabalho de cada um desses setores esteja bem feito e livre de erros, pois, juntos, os resultados obtidos dão vida à concepção do *game*.

BETHKE (2003) caracteriza de uma forma sucinta e poética a equipe de desenvolvimento de *games*:

"Pessoas criativas adoram dividir seus sonhos, pensamentos e mundos. Artistas querem lhe mostrar um mundo, Músicos querem que você sinta o mundo, Programadores querem que você experimente o mundo, e *Game Designers* querem que você esteja dentro desse mundo."

3.2.1 Game Design

É aqui que acontece a "criação" de fato do *game*, onde idéias são trabalhadas e transformadas em realidade. É preciso definir e analisar a história e se há lógica nela, a mecânica, os níveis (fases do *game*), o que pertence e o que deve ser mantido de fora, o que pode e o que não pode ser feito. *Game Designers* devem ter criatividade e sensatez para fazer com que a história mostrada em um mundo de fantasias seja convincente e interessante.

3.2.2 Código

É a programação que tira do papel a idéia de um *game*. O código dá vida ao que era apenas um rascunho, faz com que o *game* realmente aconteça. Para isso, a equipe de programação deve estar interligada ao *Game Design*, à Arte e ao Som, para seguir o que foi proposto, unir todas as partes e fazer com que o projeto funcione. Há programadores responsáveis pela aplicação de uma IA (Inteligência Artificial), determinando como componentes de dentro do *game* se comportam, programadores responsáveis por gráficos 3D e pela mecânica, precisando ter conhecimentos específicos de matemática e física, e programadores responsáveis pelo áudio.

3.2.3 Arte

É através da arte que conseguimos ver a idéia dos *Game Designers*. Eles apresentam o mundo e suas características, dão cor e forma ao que era, no início, imaginação. Artistas conceituais definem objetos, cenários e personagens criados pelos *Game Designers*. Inicialmente, são simples rascunhos, com o objetivo de obter aprovação da equipe. Depois, ganham cores e diferentes ângulos, para uma melhor análise e decisão se aquele é ou não o cenário desejado. Artistas 2D são experientes em pinturas e desenho, fazendo retratos dos personagens, desenhos de cenário, texturas para polígonos e arte de menus. Artistas 3D fazem modelagens em 3D (de objetos e personagens).

É muito importante que as equipes de *Game Design* e Arte trabalhem juntas, pois a segunda é responsável pela ambientação e pelo clima do *game* criado pela primeira.

3.2.4 Som

O som ajuda na ambientação e no clima do *game*, aumenta o grau de realidade da experiência sendo vivida. Para imergir o jogador dentro daquele universo, é essencial que haja músicas que provoquem emoções e efeitos sonoros que convençam o jogador de que ele está interagindo com aquele mundo. Músicas podem surgir de orquestras reais ou sintetizadores e softwares que simulem orquestras. Seja como for, em uma equipe de desenvolvimento de *games* deve existir um bom compositor que crie o impacto emocional exigido pela história e pelo clima proposto pelos *Game Designers*. Por isso, é muito importante que as equipes de *Game Design* e Som trabalhem juntas, pois a segunda também é responsável pela ambientação e pelo clima do *game* criado pela primeira.

3.3 Expectativas em relação a Games

"O que os jogadores esperam de um *game*? O que eles querem vivenciar e o que os faz terem interesse em jogar?" Essas são as principais dúvidas que surgem na cabeça de um *Game Designer*.

Criadores de *games* buscam produtos com diferencial, algo que nunca tenha sido feito nesta indústria. Um *game* que se destaque por sua criatividade, sua jogabilidade, seu impacto e inovação, e que tenha uma boa aceitação por todos esses aspectos.

Segundo ROUSE (2001), alguns dos aspectos que jogadores querem encontrar em *games* são:

• **Desafio**: Pessoas buscam desafios em *games*. São os desafios que, em primeiro lugar, diferenciam os *games* de outras formas de entretenimento, como livros, filmes etc (ROUSE, 2001). Esses desafios motivam jogadores a interagirem com um mundo virtual, a pensarem em diversos pontos de vista, a buscarem técnicas e estratégias diferentes para solucionar um dado problema. Nesse universo, a dificuldade e os obstáculos são os grandes responsáveis pela diversão.

Quando um obstáculo é superado, há a aprendizagem. Dessa forma, *games* que desafiam o jogador podem ser vistos como forma de aprendizado, sendo esse aprendizado limitado ao contexto do *game* ou podendo ser aplicado a aspectos da própria vida do jogador.

• **Socialização**: Pessoas gostam de *games multiplayer*[10]. Não é apenas a interação com um universo que desperta o interesse de um jogador, mas também a interação com outros jogadores: o que ele pode fazer ou não nesse universo com um grupo de pessoas, como tomar decisões e traçar estratégias contando com outros participantes. O intenso uso de *chats* (bate-papo) de texto e conversações por voz dentro dos *games*, até mesmo em *games* de ação (onde que a atenção e movimentos rápidos são exigidos ao extremo), são prova de que os jogadores querem se socializar e interagir com outras pessoas.

É por isso que *games* no estilo RPG (*Role Playing Game*, a ser definido na seção 3.3) fazem tanto sucesso. Para os jogadores, é interessante navegar por um mundo onde seu personagem interage com outros personagens (controlados por outros jogadores) de forma a criar o andamento da história, solucionar problemas e cumprir tarefas. Essa interação ajuda no andamento do *game*, porque o jogador percebe que nunca está sozinho naquele mundo.

---

[10] *Games* em que há interação com outros usuários.

Embora grande parte dos *games singleplayer*[11] tenha outros personagens virtuais que interagem com o jogador, a experiência vivida não é a mesma. Em um ambiente *singleplayer*, a personalidade dos personagens é desenvolvida através de IA – Inteligência Artificial, simulando um jogador real. Contudo, como são programadas, as IAs não apresentam a imprevisibilidade dos jogadores reais em ambientes *multiplayer*. É dessa imprevisibilidade que surge o desafio, que motiva jogadores no mundo virtual.

• **Emoção**: Pessoas querem sentir emoções. Medo, desejo de vingança, excitação, tensão, satisfação. Independentemente de serem positivas ou negativas, jogadores esperam sentir algo ao jogar um determinado *game*, que as "prenda" naquele ambiente, que as estimule e as faça insistirem. Mesmo *games* que não apresentam a possibilidade de vitória, como Skifree (Microsoft, 1991), pessoas continuam jogando com o objetivo de bater recordes, uma forma de se sentirem satisfeitos consigo mesmos.

• **Fantasia**: Pessoas gostam de fantasia. Experiências mundanas se tornam repetitivas e desinteressantes. Muitas pessoas buscam em livros, filmes e novelas um pouco de não-realidade, onde coisas excitantes acontecem, indivíduos interessantes são apresentados, mundos novos e mais atraentes surgem. Outras pessoas buscam essa experiência em *games*, sendo uma forma de escapar da realidade e se transportar para um mundo completamente diferente.

Além disso, *games* se mostram um ambiente seguro para as pessoas atuarem da maneira que bem entenderem. Como não há conseqüências reais, muitos jogadores adotam, dentro do *game*, personalidades diferentes das suas verdadeiras, vivendo uma realidade-alternativa. Dessa forma, um jogador pode atropelar pessoas ou atirar em inimigos sem que isso lhe transforme em um assassino de fato, e isso é excitante aos seus olhos.

Após um jogador decidir se aventurar em um *game*, ele terá expectativas do *game* em si. Se os resultados não corresponderem a essas expectativas, é muito provável que o jogador se canse do *game* com facilidade, e busque novas opções.

Segundo ROUSE (2001), uma vez que o jogador experimenta um *game*, alguns fatores que ele espera encontrar são:

---

[11] *Games* em que não é possível a interação com outros usuários. São usados utilitários para simular outros personagens dentro do *game*, chamados de *Robots* ou *Bot,* implementados com uma IA.

• **Um mundo consistente**: Um mundo virtual que não tenha uma lógica ou consistência confunde os jogadores. Pior que isso, é frustrante. Conseqüências imprevisíveis para ações tomadas tornam aquele universo pobre, sem sentido e até irritante.

• **Direção**: Pessoas esperam saber o que deve ser feito para vencer um *game*, e como fazê-lo. Não ter direções remete à vida real, que, muitas vezes, é do que os jogadores estão tentando escapar. Eles esperam entender qual é a meta do *game* e receber algumas instruções de como conseguir atingir essa meta.

• **Completar uma tarefa de forma incremental**: Uma vez que o jogador saiba qual é o objetivo a ser alcançado e como alcançá-lo, ele espera que isso seja feito de forma incremental, por sub tarefas a serem completadas para atingir a meta final.

• **Estar imerso naquele universo**: A falta de verossimilhança, falta de consistência e erros no código são alguns dos fatores que podem arruinar a imersão de um jogador dentro de um universo virtual. Uma vez imerso, cada vez que ele for tirado da sua fantasia, mais difícil se torna voltar a ela.

• **Falhar**: Se uma pessoa desejasse ter uma experiência simples, bastaria ler um livro. Um jogador quer vivenciar enigmas, vencer obstáculos, passar por dificuldades e se orgulhar de ter conseguido. Por isso, um *game* que não traga desafios não será interessante.

Ao falhar, o jogador aprende com seus erros. Imediatamente ele consegue identificar onde errou ou por que sua estratégia não deu certo. Tentativas posteriores serão sempre melhorias incrementais da técnica e do raciocínio do jogador.

• **Não ficar "preso" em um ponto**: Um jogador fica frustrado quando descobre que não há possibilidade de sair de um determinado ponto do *game*. Se, por exemplo, o jogador só puder passar de fase se tiver coletado um item localizado em uma sala já inacessível, e se ele não sabia que tinha que fazer isso, nada mais pode ser feito a não ser recomeçar o *game* e jogar tudo de novo. Isso tira a diversão do jogador, que terá que se repetir porque em nenhum momento foi informado o que deveria ser feito. O método de sair de um obstáculo pode ser difícil, contanto que seja possível.

• **Participar ao invés de assistir**: Qualquer método utilizado para dar instruções do *game* que seja muito longo se tornará gradualmente desinteressante ao jogador. Ele não quer assistir a vídeos longos ou textos enormes que o instruam por aquele mundo. Ao contrário, o jogador gosta de entender o que se passa e o que deve fazer simplesmente atuando no *game*, e

não ficando de braços cruzados. É muito mais interessante aprender com os próprios erros do que sendo ensinado por vários e entediantes minutos.

3.4 Gêneros de Games

Atualmente, existe uma grande variedade de gêneros de *games*. Alguns dos mais conhecidos são:

• **Ação**: Tipo de *game* que exige velocidade por parte do jogador e testa seus reflexos.

• **Aventura**: Tipo de *game* onde o jogador conduz o protagonista em uma história através de enigmas e charadas. Nesse tipo de *game*, é dada ênfase ao enredo, e não à parte artística.

• **Cartas**: Este tipo de *game* simula os games de cartas conhecidos, como Poker, Buraco, Sueca, Paciência etc.

• **Corrida**: O jogador aposta corrida com o computador ou com outros jogadores.

• **Esportivo**: Este tipo de *game* simula esportes tradicionais. O jogador pode jogar sozinho ou com um time.

• **Estratégia**: A habilidade do jogador de tomar decisões estratégicas é fundamental nesse tipo de *game*.

• **Luta**: O jogador enfrenta o computador ou outro jogador em um combate corporal.

• **Musical**: Este tipo de *game* é baseado na interação do jogador com música para pontuar.

• **Plataforma**: O jogador, assumindo um personagem, deve andar por plataformas e enfrentar obstáculos. Esse é um tipo de *game* 2D.

• **Quebra-Cabeça (Puzzle)**: Tipo de *game* que exige bom raciocínio por parte do jogador, para resolver um problema dado.

• **Role-Playing**: Tipo de *game* em que o jogador interpreta um personagem da história.

• **Simulação**: Tem a intenção de simular o mundo real ou um mundo fictício.

• **Tabuleiro**: Este tipo de *game* simula os jogos de tabuleiro conhecidos, como Damas, Xadrez, Dominó, Gamão etc.

• **Tiro**: O jogador atira livremente em outros jogadores ou em *bots* em um *game* de tiro de primeira pessoa (*First Person Shooter* – FPS) com a visão do personagem ou em um *game* de tiro de terceira pessoa (*Third Person Shooter* – TPS), onde o jogador vê seu personagem por trás.

3.5 Desenvolvimento de Games *e Metodologia Ágil*

Em se tratando de desenvolvimento de *games*, talvez a principal e mais importante característica seja o fato de que os requisitos mudem com freqüência. Eventualmente, uma funcionalidade no *game* pode não ser mais necessária ou deve ser mais trabalhada, algumas características de cenário e personagens podem ter que ser modificadas. Efeitos sonoros e músicas, por exemplo, podem ter que passar a ambientar o jogador em um clima diferente daquele determinado pelo *Game Designer* no início do projeto.

Essas constantes mudanças muitas vezes se devem à carência de profissionais na equipe com conhecimento específico de determinada área do desenvolvimento, além do fato de que empresas de *games* estão constantemente lidando com o desafio de entregarem um produto final de qualidade dentro de prazos curtos estipulados.

Como os requisitos estão em constante mudança, é uma boa opção adotar um método que saiba lidar facilmente com elas, foque na interação da equipe de desenvolvimento com o cliente e apresente resultados de tempos em tempos antes que o produto esteja concluído. Dessa forma , se houver alguma correção a ser feita, ela será tratada assim que a mudança for identificada. Dito isso, é possível imaginar que *games* desenvolvidos através de metodologia convencional apresentem mais problemas do que aqueles desenvolvidos por uma ótica ágil.

Segundo KASPERAVICIUS, BEZERRA, SILVA e SILVEIRA (2008), métodos ágeis priorizam a minimização de riscos através do desenvolvimento de software em iterações relativamente curtas, levando a uma reanálise de prioridades por parte da equipe ao final de cada iteração.

Assim, é possível observar que métodos ágeis são métodos que se adaptam às eventuais mudanças que venham a existir em um projeto. Métodos tradicionais, por outro lado, são métodos que tentam prever e se precaver em relação a essas mudanças.

Dessa forma, adotando uma metodologia ágil no desenvolvimento de *games*, as chances de o produto final não ser o que era esperado diminuem consideravelmente, já que são verificadas, a cada iteração, se as funcionalidades apresentadas estão de acordo com o especificado pelos requisitos. Se não forem, essas funcionalidades são corrigidas ou melhoradas, de forma que o erro não se propague e seja notado apenas com o projeto já finalizado.

3.5.1 Scrum

Para este trabalho, nos interessa abordar a metodologia ágil *Scrum* (SCHWABER e BEEDLE, 2001). Mais sobre a aplicação adaptada deste método no desenvolvimento de *games* será abordado no capítulo 4. Nesta metodologia, baseada em pequenas equipes, a evolução do desenvolvimento é dividida em iterações, denominadas *Sprints*, pequenos intervalos de tempo em que a equipe trabalhará visando entregar, ao final de cada prazo, uma versão cada vez mais completa do produto.

Um conjunto de tarefas é priorizado pelo *Product Owner*, membro da equipe que estabelece maior comunicação com o cliente e sabe suas necessidades. Esse conjunto de tarefas é denominado *backlog*. O *Product Owner* chega a conclusões com o cliente sobre as próximas funcionalidades que devem ser implementadas, seguindo uma ordem de importância estabelecida por ele, e então as passa para a equipe de desenvolvimento. A equipe, por sua vez, desmembra essas funcionalidades em tarefas, que deverão ser concluídas até o final do *Sprint*, sendo apresentadas ao cliente como o resultado da iteração, e mostrando o produto, agora, com mais funcionalidades.

As tarefas do *Sprint* corrente são escolhidas entre os membros de cada equipe na *Sprint Planning Meeting*, observando a possível dificuldade e o provável tempo de conclusão de cada uma delas. Tarefas que sejam muito complexas ou demoradas talvez tenham que ser decompostas em tarefas menores, de forma que seja possível terminá-las em um *Sprint*.

Ao longo do *Sprint*, há reuniões diárias de curta duração, as *Daily Meetings*, onde a equipe se reúne para discutir problemas encontrados, tirar dúvidas e decidir os próximos

passos a serem tomados para que as tarefas sejam concluídas. Imprevistos e impedimentos que atrapalhem o andamento do trabalho das equipes devem ser resolvidos pelo *Scrum Master*, responsável pelo bem estar da equipe, de forma que os desenvolvedores devam apenas se preocupar com seus trabalhos. Ao final de cada *Sprint*, na *Sprint Review Meeting*, as equipes se reúnem para discutirem suas impressões sobre a iteração, e os resultados são apresentados ao cliente.

O *Scrum* é uma metodologia válida para o desenvolvimento de *games* porque é adaptável e consegue lidar com mudanças no plano do projeto, como afirma MULLER (2009) em sua tese. Como os requisitos e as funcionalidades de um *game* em desenvolvimento estão em constante mudança, a aplicação desta metodologia se torna eficaz. Segundo o autor, "um projeto *Scrum* pode suportar grandes impactos em seu escopo, como alterações de tecnologia, equipe, prazo ou mesmo recursos financeiros".

Contudo, a metodologia *Scrum* não aborda ou explicita métodos para o processo de teste. Como a tarefa de testes, então, pode ser implantada em um ambiente cujo desenvolvimento adota uma estratégia ágil?

3.6 Qualidade dos Games

Talvez a melhor maneira de testar um *game* seja jogando. Existe, contudo, um problema nesse método: demoraria um tempo muito longo para se testar um *game* inteiro, e talvez isso nem seja possível. Por isso, enquanto a indústria não apresenta um método unificado de testes, cada projeto de *games* tem seu próprio processo para assegurar a qualidade (BETHKE, 2003).

Algumas empresas possuem uma equipe responsável pela qualidade dos *games*. Essa equipe acompanha o progresso do *game*. Seu papel é executar testes de código (para verificar se o código até então produzido possui defeitos), testes de compatibilidade de hardware e sistema operacional, testes de rede (focadas especialmente em *games multiplayer*, certificando-se de que opções de rede sejam testadas e funcionem corretamente), entre outros.

Segundo BETHKE (2003), o problema em ter uma equipe de testes por muito tempo é que sua habilidade de discernir problemas fundamentais com jogabilidade e usabilidade é comprometida quando a equipe aprende melhor o game e perde sua análise crítica de jogador.

Segundo HILL (2007), G*ame Tester* é uma profissão na área de *games* muito procurada e pouco valorizada, mas a grande vantagem de atuar nesta área é o fato de que, sendo responsável por testes, um profissional deve discutir com vários outros desenvolvedores no projeto, de todas as áreas, pois *bugs* são encontrados em toda a partes. Isso dá a um G*ame Tester* uma vasta noção do que é preciso para se fazer um *game*.

O que poderia ser uma definição de um game de qualidade? Na verdade, essa é uma questão subjetiva, já que o que leva um game a ser considrado como um bom game depende do ponto de vista de quem o avalia.

Há uma classificação dentro da comunidade de *games* que qualifica o produto. AAA é a maior "nota" que pode ser atribuída a um *game*, de forma que *Games* AAA são classificados como *games* que, além de outros aspectos, apresentam como principais pontos (JUUSO, 2006):

- Entretenimento balanceado e contínuo do início ao fim do game
- Ótima interface gráfica
- Mercado vasto
- Vendas elevadas
- Direção áudio-visual polida
- Execução técnica e artística perfeita
- Testado exaustivamente
- Livrre de defeitos
- Orçamento alto

Ainda assim, é difícil definir exatamente o que é um *game* de qualidade. São abertos debates sobre o que um *game* deve possuir pra ser considerado de qualidade, e se ele é ou não necessariamente bom e divertido.

Um consenso geral sobre um *game* de qualidade diz que o produto deve entreter, divertir, desafiar e ser livre de defeitos, de forma que as duas partes que compõem um *game*, técnica e humana, estejam bem trabalhadas e integradas, proporcionando ao jogador a experiência proposta.

3.7 Considerações Finais

Neste capítulo, foram vistas as partes que compõem um game e suas responsabilidades no desenvolvimento para que o produto final tenha as exigências citadas. É possível perceber que essas exigências serão cumpridas se as equipes trabalharem em conjunto, mantendo uma contínua integração entre essas partes.

Como métodos ágeis são adaptáveis a mudanças em um projeto, viu-se que o desenvolvimento de um *game* através de uma metodologia ágil apresenta vantagens para o processo, aumentando, conseqüentemente, a qualidade do produto.

No próximo capítulo, será apresentada uma abordagem de testes para a área de *games*, de forma a minimizar as dificuldades encontradas na integração dessas duas áreas e aumentar a qualidade do *game*.

4. Testes aplicados em *Games* desenvolvidos através de uma Metodologia Ágil

4.1 Introdução

Como base para este trabalho, foi acompanhado o trabalho diário de uma equipe de desenvolvedores que, atualmente, está trabalhando em um *game* que deve ser lançado para o console de jogos Xbox 360. Foram levadas em consideração as informações e as estratégias usadas por essa equipe, que faz parte da empresa Aiyra, e se encontra dividida conforme mostrado no capítulo 3, seção 3.2.

O *game* em desenvolvimento está sendo feito por essa equipe através da metodologia ágil *Scrum4Games*, adaptada da metodologia *Scrum* por MULLER (2009). Embora o presente trabalho aborde apenas o cenário com esta metodologia adaptada, é interessante notar alguns dos principais pontos de contraste entre o gerenciamento de *games* e a metodologia *Scrum*, para uma melhor compreensão da necessidade de adaptação.

Nas seções a seguir, será vista, além de especificações do *game* em desenvolvimento citado nesta monografia, a relação entre o gerenciamento de *games* e a metodologia ágil *Scrum*, detalhando os problemas decorrentes dessa interação e as adaptações feitas. Será também vista uma abordagem de testes para o *game* em desenvolvimento, focando-se nesta metodologia adaptada.

4.2 Game *em desenvolvimento*

O *game* que está sendo desenvolvido pela equipe estudada baseia-se no gênero Plataforma, como visto na seção 3.4. É um *game* 2D em que duas personagens lutam pela posse de um determinado objeto em cena, com o objetivo de alcançá-lo e mantê-lo seguro da equipe rival.

Para conseguir alcançar o objetivo do *game*, as personagens possuem objetos que auxiliam na movimentação e servem como ataque a outras personagens e alcance para o objeto desejado (chamado *Token*), que se encontra protegido por uma personagem maior e mais forte.

A equipe sofreu modificações ao longo do desenvolvimento, contendo, desde o início do projeto, um total de 6 artistas, 6 programadores, 3 *game designers* e 1 músico, que trabalharam com foco na interação entre essas partes.

No código, foram utilizados o framework XNA (MICROSOFT, 2010) cuja linguagem de programação é C#, específico para jogos desenvolvidos para Xbox, e a engine Torque X 2D (GARAGE GAMES, 2010), além do Microsoft Visual C# (MICROSOFT, 2010), um ambiente integrado de desenvolvimento. Para o controle de versões dos trabalhos de todas as equipes, foi utilizado o Subversion (detalhado na seção 4.5.2). Atualmente, o *game* encontra-se na fase de testes.

4.3 Scrum *x Gerenciamento de* Games

O *Scrum* é, em sua essência, uma metodologia ágil para gerenciamento de projetos de software. Embora um *game* seja considerado também um software, possui certas peculiaridades que dificultam a aplicação desta metodologia no gerenciamento do desenvolvimento de *games*.

Tendo como base o trabalho de MULLER (2009), problemas foram observados nesta ligação entre o *Scrum* e o desenvolvimento de um game. Seguem alguns deles:

- **Ausência de documentação do *Scrum* x Documento de *Game Design***

Grande parte da agilidade deste método se deve à ausência de documentação. Porém, no desenvolvimento de *games*, o Documento de *Game Design* tem um papel fundamental na organização das funcionalidades do produto. Por conter não apenas aspectos técnicos, mas também dados mais subjetivos como clima de *game* e mensagens que devem ser passadas ao jogador (fundamentais principalmente às equipes de arte e som), este documento é indispensável durante o desenvolvimento.

▪ **Foco na *Daily Meeting* x Interesse na discussão de *Game Design***

É muito importante que os desenvolvedores mantenham o foco nesta reunião, que deve ser curta e objetiva (geralmente, 15 minutos no máximo), de forma a deixar o resto da equipe a par das tarefas que estão sendo realizadas e as dificuldades e dúvidas encontradas.

No desenvolvimento de um *game*, entretanto, é comum o interesse dos desenvolvedores em criticar e dar opiniões sobre determinados aspectos. Com isso, o que era o papel do *Game Designer* torna-se, então, uma discussão entre membros das equipes, que trocam idéias para buscar novas e melhores funcionalidades a serem implementadas. Essas discussões estão muito presentes nas *Daily Meetings*, onde cada membro da equipe tem um tempo para falar o que fez e o que pretende fazer.

Neste momento, os membros tendem a dar suas opiniões do que facilitaria ou "seria melhor" para o seu trabalho. Com réplicas de outros participantes, a reunião tende a perder o foco e durar mais do que deveria, a princípio.

Embora essas críticas sejam, segundo MULLER (2009), de extrema importância para a melhoria do produto final, deve-se tomar cuidado em não deixar que essas discussões surjam em momentos errados.

▪ **"Fechamento" do *Game Design* durante o processo de desenvolvimento**

Como há pouco visto, é improvável que a equipe de desenvolvimento não tenha idéias novas e diferentes das formuladas pelo *Game Design* e queira compartilhá-las. Esse é um aspecto importante para o desenvolvimento de um *game*, que só tem a ganhar com as novas possibilidades abordadas. De fato, quanto mais idéias forem compartilhadas e bem organizadas, melhor as chances de se obter um *game* divertido e inovador.

Além de novas idéias, surgem imprevistos. Se determinada funcionalidade não puder ser implementada a tempo para a entrega do *game*, ou se exige conhecimentos específicos que os membros de uma equipe não possuem, essa funcionalidade terá que ser adaptada ou retirada do projeto. Isso pode modificar o *game*, não só em detalhes, mas também como um todo.

Muitas das modificações surgem da experiência que a equipe vai adquirindo ao longo do desenvolvimento. Por isso, não é recomendável que o *Game Design* seja definido e considerado como pronto. Obviamente, a equipe precisa de decisões bem definidas para um ponto de partida (tendo o mínimo de informações para o início do trabalho), mas com o passar

do tempo, o *Game Design* deve estar aberto a novas idéias, com possibilidade de modificações.

▪ **Interrupção do trabalho criativo para a *Daily Meeting***

Como já visto, *games* são criados por partes técnicas e partes criativas. Principalmente com membros da equipe de *Game Design*, Som e Arte, a criatividade é um fator fundamental para que o *game* possa não apenas ser bom, mas passar as emoções e o clima corretos para o jogador.

Por isso, é importante que todo o trabalho criativo sendo feito seja completado ou, na pior das hipóteses, interrompido em um ponto onde o membro da equipe se "encontre" ao retornar para ele, antes de começar a participar da *Daily Meeting*. Se houver interrupção em um ponto qualquer, talvez o artista não consiga retornar ao raciocínio original, o que pode comprometer horas de trabalho.

Por esses motivos, talvez não seja aconselhável a aplicação do *Scrum* no desenvolvimento de um *game* sem que haja uma adaptação desta metodologia (MULLER, 2009), de forma que as complicações identificadas sejam minimizadas.

4.4 Scrum *adaptado para* Games*: Cenário de estudo*

Tendo em vista os problemas citados na utilização da metodologia *Scrum* no desenvolvimento de g*ames*, algumas adaptações precisaram ser feitas, de forma que as desvantagens dessa implantação fossem minimizadas.

Tais adaptações, propostas por MULLER (2009), compuseram a metodologia *Scrum4Games*. Esta metodologia foi adotada pela equipe no processo de desenvolvimento do *game* estudado neste trabalho.

4.4.1 Game Designer Master

No trabalho que apresenta o *Scrum4Games*, o autor afirma que *Game Designers* são "profissionais altamente associados ao valor do negócio e que compreendem o fato de que limitações tecnológicas podem influenciar na modelagem inicial do produto definida por eles." Com isso, é fácil perceber que estes profissionais estão na posição de analisar as funcionalidades pelo que elas agregam ao *game*, e não de quais formas elas foram implementadas. Os *Game Designers* dão valor a essas funcionalidades e desejam vê-las sem problemas ao final do projeto, de forma que o que tenha sido definido pela sua equipe seja validado.

Por isso, considerar os *Game Designers* como "clientes" é uma abordagem válida, já que ambos os papéis estão essencialmente preocupados com o produto final. Em outras palavras, este profissional simplesmente dita os objetivos que o produto deve alcançar.

Dessa forma, o *Scrum4Games* apresenta o responsável pela equipe de *Game Design* desempenhando o papel de *Product Owner* da equipe de desenvolvimento[12].

4.4.2 Game Design Wiki

Como uma de suas características ágeis, o *Scrum* dita a redução de documentação. Isso gera conflitos se esta metodologia for aplicada ao desenvolvimento de *games*, já que a documentação é essencial para esta última por conter aspectos técnicos e gerais, que ajudam a organizar e entender o *game*.

Por isso, uma forma eficiente de comunicação entre as equipes de desenvolvimento foi a chamada *Game Design Wiki*. Essa documentação apresenta aos desenvolvedores as decisões da equipe de *Game Design*, bem como eventuais modificações e adaptações nas funcionalidades do produto.

---

[12] O *Scrum4Games* apresenta uma abordagem diferente para a divisão de equipes. Nesta metodologia, a equipe de *Game Design* é separada da equipe de desenvolvimento (Código, Arte e Som). Assim, ambas as equipes possuem um *Scrum Master*. Na equipe de *Game Design*, este responsável é denominado *Game Designer Scrum Master*, que desempenha o papel de *Product Owner* da equipe de desenvolvimento.

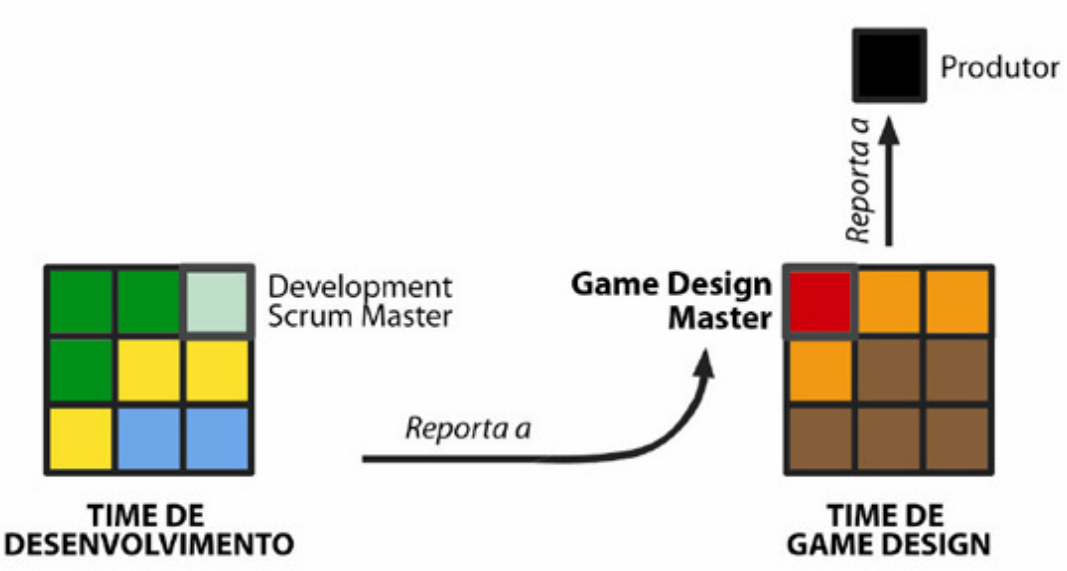

Figura 4 – Composição completa dos papéis do *Scrum4Games* (MULLER, 2009).

Assim, as equipes poderiam tirar eventuais dúvidas que surgissem sem que, para isso, tivessem que interromper o trabalho dos *Game Designers*, do *Scrum Master* ou até mesmo dos membros das outras equipes. Um membro da equipe de arte, por exemplo, poderia colocar na *Wiki* uma dúvida não tão urgente para a equipe de código, perguntando se era possível determinada integração entre elas.

Além disso, o *Game Design Wiki* funciona também como um canalizador de idéias e opiniões. Cada seção do documento possui um espaço aberto para sugestões e críticas por parte dos membros da equipe, o que não só leva os desenvolvedores a analisarem suas próprias áreas, como também as áreas das outras equipes. As sugestões e opiniões são então anotadas e levadas à *Weekly Game Design Meeting*, explicada a seguir.

4.4.3 Weekly Game Design Meeting

Foi observado um grande interesse por parte dos membros das equipes no desenvolvimento do *game*. Discussões para a melhoria de aspectos específicos e gerais do game eram freqüentes, principalmente nas *Daily Meetings*.

Contudo, era de extrema importância se ater ao foco da reunião, que se propunha a apenas comunicar ao resto da equipe o que cada membro desenvolveu em um dia, dentro da sua área, além das dificuldades encontradas. Mesmo assim, era fácil para os desenvolvedores

se deixarem levar pelo clima de discussão, fazendo com que a reunião abrangesse mais assuntos do que deveria.

Com isso, foi feita a *Weekly Game Design Meeting*, que acontecia ao final de cada semana de trabalho, com o intuito de coletar opiniões e sugestões dos membros participantes, não apenas na área de *Game Design*, mas também nas áreas de Som e Arte. O Código também estava aberto a discussões, mas como essa área do desenvolvimento é muito mais técnica e exige muito menos "imaginação e criatividade", não havia muitas sugestões a serem anotadas.

A *Weekly Game Design Meeting* tinha como objetivo analisar e discutir as sugestões e/ou críticas apontadas pelos membros da equipe na *Game Design Wiki*, de forma que todas as opiniões eram ouvidas e comparadas. Isso fez com que o aprendizado da equipe crescesse exponencialmente, já que cada membro agora se via imerso nas outras áreas do desenvolvimento, discutindo aspectos específicos. A compreensão de dificuldades de integração entre essas áreas foi aumentada, de forma que possíveis dúvidas futuras fossem extinguidas.

Além dessa vantagem, foi observado que a motivação dos desenvolvedores cresceu substancialmente, já que eles podiam expor suas idéias, com a possibilidade de serem aceitas e integradas ao *game*. Ainda, foi vantajoso à equipe de *Game Design* ter opiniões tanto técnicas como gerais a respeito dos assuntos abordados, o que contribuiu para a diversidade de idéias para a formulação de jogabilidade e de clima do *game*.

4.5 Tarefas de Testes

Como visto no capítulo 3, as tarefas de testes no desenvolvimento de um *game* se baseavam no trabalho do *Game Testers*, onde os aspectos gerais eram analisados quando o produto era executado. Obviamente, como um *game* é, antes de mais nada, um software, são feitos também testes que verificam o bom funcionamento das funções do código (afinal, se isso não acontecesse, o *game* sequer seria executado). O *Scrum4Games*, contudo, não discute ou desenvolve o assunto de testes em sua abordagem, o que limita essa adaptação para o desenvolvimento de *games*.

Será abordado a partir de agora uma forma que especifica as tarefas de testes não só como descrito anteriormente, mas integradas entre si. Desta forma, as tarefas dos *Game*

Testers e dos testadores de código serão analisadas em conjunto, de forma a apresentar melhores resultados para o produto.

Foram feitos três tipos de testes: Testes pela equipe de desenvolvimento, testes por *Game Testers* e testes feitos pela equipe na ótica de *Game Testers*. Veremos as vantagens trazidas por essas três abordagens distintas para a qualidade do *game*.

4.5.1 Processo Proposto

Os testes realizados pela equipe de desenvolvimento abrangem Testes de Unidade nas partes específicas de produção (*Game Design*, Arte, Som e Código) e Testes de Integração, realizados gradualmente entre essas partes. Os testes realizados por *Game Testers* e pelos desenvolvedores com a perspectiva de *Game Testers* compõem os Testes de Sistema, e os Testes de Aceitação são compostos pelos Testes Alfa[13] e Beta[14].

Desta forma, essa abordagem complementa o processo de testes comumente realizado no desenvolvimento de *games*. A figura 5 detalha o processo dentro desta abordagem, As contrubuições deste trabalho estão destacadas em amarelo e rosa.

Nas próximas seções, serão explicados e detalhados cada um dos testes, discutindo a importância de todos eles.

---

[13] Testes conduzidos pelo usuário no ambiente do desenvolvimento, com o desenvolvedor registrando problemas e defeitos encontrados.

[14] Testes conduzidos pelo usuário em suas próprias instalações. Neste tipo de teste, o desenvolvedor não está presente, e o usuário anota os problemas encontrados para, posteriormente, relatá-los.

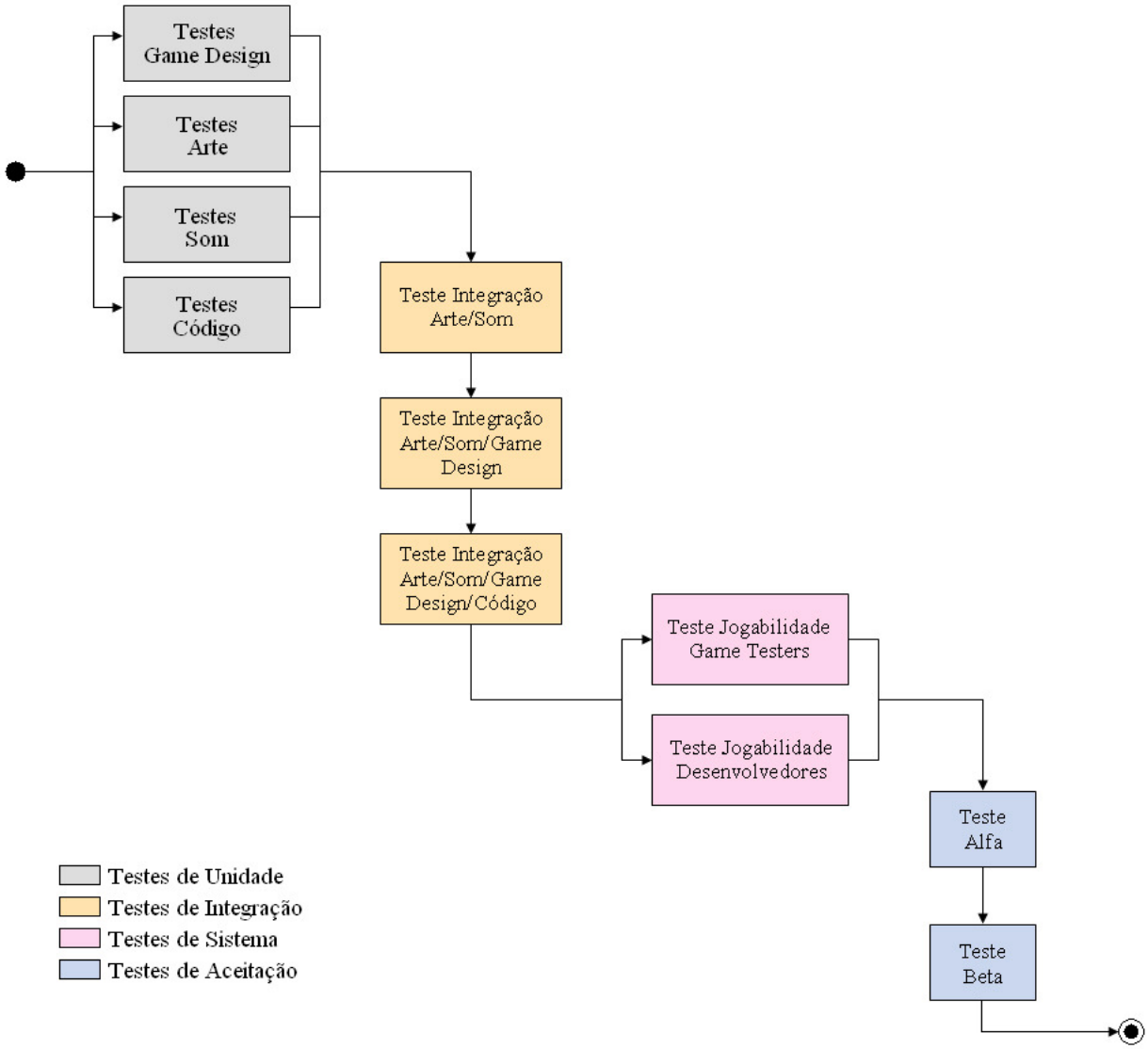

Figura 5 – Processo de Testes para *Games*.

4.5.2 Testes na equipe de desenvolvimento

Como dito anteriormente, as especificações de um *game* estão em constante modificação. Eventualmente, algum requisito se apresenta impossível de ser implementado e, por isso, é modificado para se adaptar às restrições, podendo ser até mesmo banido do *game*. Uma funcionalidade pode, então, ser completamente modificada enquanto a equipe de desenvolvimento se empenha para terminar suas tarefas no meio do S*print* corrente.

Na melhor das hipóteses, essa modificação não trará necessariamente problemas para todas as equipes. Porém, foi observado que, normalmente, quando um requisito é reavaliado e reajustado, a integração entre as equipes do desenvolvimento é afetada.

Por isso, notou-se a importância de se aplicar testes a todo o momento do desenvolvimento, tentando captar falhas e erros de integração entre as diferentes áreas do desenvolvimento do *game*. Testes mostraram diversos problemas na integração entre Arte e Som, Arte/Som e Game Design, Arte/Som/Game Design e Código e conflitos internos na equipe de Código.

▪ Testes na integração entre Arte e Som

Não era possível a essas duas equipes trabalharem sempre juntas. Eventualmente, no início do projeto, quando arte e som conceituais estavam sendo apresentados como produto das informações do *Game Design*, as duas equipes interagiam de forma intensa, analisando a melhor forma de integrar suas criações e manter um "mundo" coeso dentro daquela estória.

Com a aprovação do material inicial, essas duas equipes começaram a se empenhar especificamente em suas áreas. Contudo, modificações no aspecto do *game* eram constantes. Por motivos de adaptação em partes do *Game Design* ou do Código, os trabalhos da Arte e do Som tinham que ser refeitos e re-integrados.

Isso não acontecia somente nos trabalhos de Som e Arte conceituais. Quando um objeto era adicionado ou modificado, as equipes tinham que analisar quais eram as melhores formas de trazer aquele objeto à vida, sempre tendo em mente que a integração dessas duas áreas deveriam torná-lo verossímil e convincente dentro daquele mundo.

Dessa forma, essas duas equipes trabalhavam não só em suas áreas de atuação, mas exerciam também um papel de testadores quanto ao clima e à ambientação do *game* sobre um trabalho único, fruto da integração de suas áreas.

▪ Testes na integração entre Arte/Som e *Game Design*

As decisões de mudança na jogabilidade, na estória ou no clima do *game* pelo *Game Design* refletiam diretamente nos papéis dos membros das equipes de Arte e Som. Com modificações naquele "mundo", essas equipes tinham que também mudar alguns aspectos específicos, ou até mesmo criar novos conceitos.

Como a ambientação no *game* é um fator importante para que ele seja considerado bom, essas equipes trabalharam juntas durante um longo tempo, tomando decisões em conjunto, de forma que o resultado final fosse satisfatório.

Eram comuns situações em que o *Game Designer* era alertado por um artista ou um músico para as dificuldades ou as desvantagens de uma mudança em particular. Assim, era importante que os membros presentes na discussão chegassem a um consenso. Obviamente, as opiniões do *Game Designer* tinham mais peso, dado o maior conhecimento deste na formulação de um *game*. Ainda assim, as intervenções das outras equipes eram válidas, já que elas também tinham uma visão importante do que poderia ser bom ou ruim para a representação do *game*.

▪ **Testes na integração entre Arte/Som/*Game Design* e Código**

A equipe do Código, durante o desenvolvimento, notava com freqüência que determinada funcionalidade deveria ser adaptada ou totalmente modificada, para que o *game* pudesse funcionar corretamente.

Era comum isso acontecer depois de algumas adaptações no *Game Design*, que ditava as regras para as outras equipes. Quando isso acontecia, muitas vezes os trabalhos das equipes de Arte e Som relacionados àquela funcionalidade já estavam concluídos. Havia então a necessidade de retrabalho dessas duas equipes, de forma a adaptar o *game* às modificações feitas pelo *Game Design*.

Porém, também eram comuns modificações feitas no Código sem que estas modificações fossem ditadas pelo *Game Design*. Quando certas funcionalidades tornavam-se impossíveis de ser implementadas (por falta de conhecimento específico da equipe de Código ou por falta de tempo para a entrega do produto), elas eram simplesmente retiradas do *game*.

Algumas dessas funcionalidades, quando banidas, não apresentavam problemas sérios para o produto como um todo – simplesmente não faziam mais parte daquele mundo, e eram tratadas como opções que nunca foram consideradas antes. Outras funcionalidades, porém, modificavam a própria jogabilidade e mecânica do *game*. Com isso, elas não poderiam ser simplesmente retiradas sem que trouxessem conseqüências fortes para o produto em desenvolvimento.

Fazia-se então necessária uma total reavaliação e reformulação do *game* por parte do *Game Design*, que precisava achar opções para a solução do problema. Juntamente com a reformulação do *game*, vinha a necessidade de um retrabalho por parte das equipes de Som e Arte, que muitas vezes tinham que começar novamente com trabalhos conceituais, de forma a adaptá-los às mudanças feitas.

Isso fazia com que pudessem surgir problemas também com os novos trabalhos da Arte e do Som integrados ao Código. Eventualmente, a "nova arte" não podia ser implementada por restrições nas funções dentro do código, ou algum "novo som" não podia ser tocado da maneira que o "novo *game design*" pedia.

Era preciso que o *Game Design* e o Código estivessem em sintonia para que o *game* funcionasse, em primeiro lugar. Qualquer problema na integração entre essas áreas colocava em risco não só a qualidade, mas o próprio funcionamento do produto final. Com isso, eram também feitos testes na integração entre o Código e os elementos das outras áreas mais subjetivas do desenvolvimento (*Game Design* e, conseqüentemente, Arte e Som).

▪ **Testes na equipe de Código**

Dentro da equipe de código, fizeram-se necessários testes mais técnicos e formais. Como eventuais alterações poderiam afetar a integridade do código – afetando, assim, o funcionamento do *game* – os testes tiveram que ser executados e acompanhados com uma maior atenção.

Além dos Testes de Unidade, que verificavam se classes e métodos funcionavam corretamente, os Testes de Regressão se mostraram fundamentais, pois analisavam se alterações em determinada parte do código afetavam apenas as partes que deveriam afetar, mantendo intactas as funcionalidades que não deveriam sofrer impactos.

Era comum pequenas tarefas de implementação serem divididas pelos membros dessa equipe. Contudo, muitas vezes, alterações por parte de um dos membros em uma parte do código afetava diretamente o trabalho de outro membro.

Para auxiliar não apenas na integração entre as partes do código, mas também no controle de versões (do código e das outras áreas do *game*), foi utilizado pela equipe o Subversion (COLLINS-SUSSMAN, FITZPATRICK e PILATO, 2008), uma ferramenta de versionamento que mantém organizadas as versões dos documentos durante o

desenvolvimento de um software, bem como o histórico de modificações comentado e explicado pelos próprios desenvolvedores.

Assim, se os testes de regressão retornassem um problema, era mais fácil para a equipe de Código solucionar esse problema, tendo em mãos todo o histórico de modificações do código do *game*.

4.5.3 Testes por *Game Testers*

Durante a fase de testes da jogabilidade do *game* em questão, os *Game Testers* deram importância a alguns aspectos apontados pela própria equipe de desenvolvimento. Dessa forma, os membros de cada equipe poderiam receber *feedback* de um aspecto específico do *game* por parte do testador que ficasse responsável por ele.

Assim, foram estabelecidos graus de importância a certos aspectos do *game* em si. Cada equipe apontava para o(s) aspecto(s) que queria que fosse(m) mais exaustivamente testado(s) pela equipe de testes, por saberem ou intuírem que poderiam ser encontrados erros nele(s).

Essa intuição vinha da experiência do próprio desenvolvimento. Assim, um desenvolvedor que tinha conhecimento do grau de dificuldade de uma determinada funcionalidade implementada exigia testes mais completos para ela. Se certa funcionalidade, no momento da codificação, por exemplo, exigisse muito da equipe de programadores, seria provável ou esperado que ela apresentasse problemas também na hora da execução do *game*, principalmente com todos os elementos (som, arte e código) em conjunto.

Além dos aspectos apontados pelos desenvolvedores, os *Game Testers* seguiram também algumas regras básicas (VIDEOGAMETESTERHUB.COM, 2009):

- **Foco no trabalho**

É muito fácil se perder dentro do ambiente que o *game* proporciona, mas é extremamente importante que o *Game Tester* saiba o que deve fazer. Seu objetivo não é ganhar do oponente ou passar de fase, mas sim checar cada opção que o *game* oferece e observar os detalhes em sua volta.

Alguns dos pontos[15] que os *Game Testers* levaram em consideração ao testar o *game*:

▫ A arte era fiel ao que se propunha? Havia alguma falha de renderização ou colorização?

▫ A música e os efeitos sonoros davam a ambientação certa? Havia algum ruído que não deveria existir?

▫ A movimentação do personagem estava fluida?

▫ A mecânica dos espaços do *game* estava correta? As paredes e o chão apresentavam alguma falha?

▫ Os objetos se comportavam de maneira correta?

▫ A jogabilidade era previsível? Era confusa?

▫ As opções de jogo funcionavam corretamente?

▫ Os sons e as animações apresentavam comportamento correto e sincronizado com as ações dadas por comandos?

▪ Uso exaustivo da técnica do "E se...?"

Um dos trabalhos do *Game Tester* é ousar. Não só achar erros, mas também tentar provocá-los, forçar situações extremas e incomuns, saindo dos padrões impostos pelo *game*. O *Game Tester* deve pensar que é o seu papel achar e causar defeitos em tudo que foi feito pelos desenvolvedores. Para isso, alguns pontos foram levados em consideração pelos testadores:

▫ O que acontece se algo que não deveria ser feito dentro do *game* é feito?

▫ O que acontece se outros botões do *game* são acionados?

▫ O que acontece se uma ação é tomada várias vezes seguidas dentro do *game*?

▫ O que acontece quando não é feito nada por muito tempo?

---

[15] Os pontos listados foram exemplos a serem considerados em geral. Alguns dos pontos mais específicos, com uso de uma *checklist* por parte da equipe de testes, serão listados posteriormente, ainda nesta seção.

- **Tomar notas**

Como os testes apresentaram muitas informações de jogo aos *Game Testers*, eles tinham a tarefa de sempre escreverem absolutamente tudo o que fosse observado de defeituoso ou simplesmente anormal no *game*. Através destas anotações, posteriormente seria feito um relatório de erros e observações por parte da equipe de testes, sendo entregue à equipe de desenvolvedores.

- **Precisão ao reportar problemas encontrados**

Foi exigido da equipe de testes que os erros reportados fossem detalhados ao máximo, de forma que o relatório gerado à equipe de desenvolvimento pudesse informá-la com precisão do que aconteceu durante os testes. Assim, os desenvolvedores conseguiriam reproduzir o erro encontrado, de forma a corrigi-lo com mais facilidade. Alguns dos detalhes que foram explicitados:

- O que exatamente estava errado?
- O que deveria acontecer, e o que de fato aconteceu?
- Qual era o impacto do erro para o *game*? Era mínimo ou importante?
- Como o *Game Tester* chegou ao ponto de erro no *game*? O que ele fez para que isso acontecesse?
- Quanto tempo se passou até o erro ser observado?
- Quantas vezes o *Game Tester* teve que tentar para que finalmente o encontrasse?
- Quantas vezes o erro aconteceu? Sempre ou apenas algumas vezes?
- O erro ocorreu sozinho ou foi conseqüência de um conjunto de ações do jogador?

- **Não testar o *game* por horas seguidas sem interrupções**

Dois aspectos foram observados inicialmente na equipe de testes. Após horas testando o mesmo *game*, observando detalhes e realizando ações repetidas, os *Game Testers* apresentavam uma das duas respostas: Ou estavam demasiadamente cansados, ou se

"perdiam" dentro do *game*. A primeira fazia com que o testador perdesse seu senso crítico e deixasse passar erros que seriam facilmente detectados em outra situação. A segunda fazia com que o testador simplesmente parasse de testar o *game*, e começasse a jogá-lo sem perceber. Por isso, a equipe de testes sempre fazia pausas para descanso entre as horas de teste, de forma a manter o foco no trabalho.

▪ **Opinião pessoal**

Foi proposto aos *Game Testers* que eles dessem suas opiniões pessoais quanto ao *game*, e reportassem suas sugestões caso tivessem alguma idéia que o tornasse mais interessante, divertido ou desafiador, ou achassem que algum aspecto como a jogabilidade, a arte ou o som pudesse ser melhorado. É muito importante a opinião de quem está de fora do desenvolvimento em si.

Seguindo essas regras básicas para a otimização da análise do *game*, os *Game Testers* foram questionados quanto a aspectos gerais (em cada área do desenvolvimento) e específicos. Tarefas de teste do código, em particular, não foram especificadas para os testes pelos *Game Testers* por ser o Código, por si só, a própria integração funcional entre Arte, Som e *Game Design*.

● Aspectos Gerais – ARTE

▪ Plano de fundo

▫ Colorização: Cores fortes? Fracas? Incomodam o jogador? Chama mais atenção do que o primeiro plano?

▫ Sombreamento: Corresponde à iluminação?

▫ Iluminação: Convincente?

▫ Desenho: Está bem feito? Bem detalhado? Tem falhas? As linhas seguem o mesmo padrão?

▫ Clima: Tem ligação com o tema do *game*? Passa ao jogador a imagem que deveria passar? Possibilita a imersão no *game*?

▪ **Personagens**

▫ Desenho: Detalhes como juntas, curvas e saliências estão bem feitos?

▫ Colorização: Cores fortes? Fracas? Incomodam o jogador? Tiram a atenção do plano de fundo?

▫ Sombreamento: Corresponde à iluminação?

▪ **Animações**

▫ Animação das Personagens: É bem feita? É fluida? Convincente? Corresponde bem às ações? Muito rápida ou muito devagar? Demora a acontecer, depois de dados os comandos?

▫ Animação de outros objetos em cena: É fluido? Convincente? "Confortável" ao observar? A movimentação corresponde à realidade?

▪ **Arte do menu**

▫ Desenho: Está bem feito? Bem detalhado? Tem falhas? As linhas seguem o mesmo padrão?

▫ Animação dos botões: Animações convincentes ao clicar ou passar com o ponteiro por cima?

▫ Colorização: As cores destoam do resto do *game*? Qual a relação entre os botões e a tela de menu?

▫ Transição entre as telas de menu: Há demora na transição? É muito rápido aos olhos? É “fluido”?

▫ Clima: Tem ligação com o tema do *game*?

● Aspectos Gerais – SOM

▪ **Sons do *Game***

▫ Músicas do *Game*: Proporcionam uma boa ambientação? Os instrumentos são bem trabalhados? Possibilitam a imersão no *game*? Loops são bem feitos? Em que

aspectos as músicas são diferenciadas? Essa diferenciação é positiva ou negativa? Como é a transição de uma música a outra?

▫ Efeitos Sonoros: Correspondem à realidade? É compatível com as ações e gestos? São sincronizados com os comandos (demoram a ser ouvidos depois do comando)? Algum dos efeitos sonoros agradam ou desagradam demasiadamente?

▪ Sons do Menu (Telas fora do *Game*)

▫ Músicas de Menu: Proporcionam uma boa ambientação? Os instrumentos são bem trabalhados? O clima passado é o proposto, dependendo da tela apresentada? Loops (quando presentes) são bem feitos? Como é a transição de uma música a outra?

▫ Efeitos Sonoros: São sincronizados com os comandos (demoram a ser ouvidos depois do comando)? Algum dos efeitos sonoros agradam ou desagradam demasiadamente?

● Aspectos Gerais – *GAME DESIGN*

▪ Clima

▫ A história é interessante? O clima é adequado? Há exageros em algum aspecto?

▪ Jogabilidade

▫ Plataformas: São de fácil acessibilidade? A quantidade de plataformas deve ser reduzida ou aumentada? Ajudam ou atrapalham os movimentos do jogador? Estão bem localizadas?

▫ Movimentos: Os movimentos são razoáveis? O jogador se adapta facilmente quando mudanças nos movimentos acontecem? Essa mudança tem pontos positivos/ negativos em que lugares específicos?

▪ **Diversão**

▫ Quão desafiador é o *game*? Quão divertido? Quão empolgante? Quão frustrante? O *game* seria mais divertido se jogado com mais pessoas? Seria mais divertido se jogado online? Qual a faixa etária que mais provavelmente se interessaria pelo *game*?

● **Pontos específicos de integração a serem analisados**

Pontos específicos foram testados fazendo-se uso de uma *checklist*. Alguns desses pontos foram considerados mais importantes ou potencialmente mais problemáticos, tornando-se necessário que todos os *Game Testers* verificassem com mais cuidado esses aspectos. Os pontos especificados foram:

Arte

□ Detalhes como juntas, curvas e saliências das personagens estão bem feitos?

□ A colorização das personagens está bem feita? As cores estão muito forte/muito fracas?

□ A colorização das personagens tira a atenção do plano de fundo?

□ As personagens estão do mesmo tamanho?

□ As personagens comportam-se de forma correta ao cruzarem na tela?

□ As iluminações e os sombreamentos das personagens e do Guardião são coerentes com o ponto de luz?

□ Os movimentos das juntas da personagem são fluidos?

- ☐ As animações das personagens são muito rápidas/devagar?
- ☐ As animações das personagens são convincentes?
- ☐ O plano de fundo chama mais atenção que o primeiro plano?
- ☐ As cores usadas no plano de fundo são muito fortes/muito fracas?
- ☐ A iluminação do cenário é convincente?
- ☐ O cenário passa o clima proposto?
- ☐ O cenário possibilita a imersão no jogo?
- ☐ As personagens se comportam de forma correta em relação ao relevo do cenário (rampas, beiradas, paredes, teto etc)?
- ☐ Os desenhos do menu estão bem feitos e bem detalhados?
- ☐ Todas as linhas dos desenhos do menu seguem o mesmo padrão?
- ☐ Animações de botão são convincentes ao clicar ou passar com o ponteiro por cima?
- ☐ As cores do menu destoam do resto do game?
- ☐ A transição entre as telas de menu é muito rápida/muito lenta?
- ☐ A transição entre as telas de menu é fluida?
- ☐ O clima passado pela arte tem ligação com o tema do game?

Som

- Músicas do jogo e do menu proporcionam uma boa ambientação?
- Os instrumentos das músicas de menu e do jogo são bem trabalhados?
- As músicas possibilitam imersão no game?
- O clima passado pela música é o proposto, dependendo da tela apresentada?
- Loops (quando presentes) são bem feitos?
- Como é a transição de uma música a outra?
- A diferenciação entre as músicas é positiva ou negativa?
- Efeitos sonoros correspondem à realidade?
- Efeitos sonoros são compatíveis com as ações e gestos?
- Efeitos sonoros são sincronizados com os comandos (demoram a ser ouvidos depois do comando)?
- Os sons são executados sempre que é dado um comando?
- Os sons são executados corretamente mesmo com comandos repetidos rapidamente?
- Algum dos efeitos sonoros agrada ou desagrada demasiadamente?

- Sons e músicas se misturam e geram confusão (cacofonia)?
- O volume dos efeitos sonoros dentro do game está muito alto/ muito baixo?
- O volume das músicas dentro do game está muito alto/muito baixo?
- O volume dos efeitos sonoros fora do game (telas de menu) está muito alto/ muito baixo?
- O volume das músicas fora do game (telas de menu) está muito alto/muito baixo?
- Os efeitos sonoros estão todos no mesmo volume?
- As músicas estão todas no mesmo volume?
- Algum efeito sonoro deve destoar dos outros?
- A relação dos volumes dos efeitos sonoros e das músicas está boa?
- Algum destes deveria ser aumentado ou diminuído?

Game Design

- A história é interessante?
- O clima passado é o clima proposto pela história?
- Há exageros em algum aspecto do clima?
- Os movimentos são razoáveis?

- O jogador se adapta facilmente quando mudanças nos movimentos acontecem?
- Essa mudança tem pontos positivos ou negativos em que lugares específicos?
- As plataformas são de fácil acessibilidade?
- As plataformas são de fácil identificação?
- A quantidade de plataformas deve ser reduzida ou aumentada?
- As plataformas estão bem localizadas?
- O game seria melhor se fosse mais rápido?
- O game seria melhor se fosse maior?
- O game seria melhor se jogado com outras pessoas?
- O game seria melhor se jogado online?
- O game desafia o jogador?
- O game é divertido? É empolgante? É frustrante? É cansativo?
- Qual a faixa etária que mais provavelmente se interessaria pelo game?

Código

- Como está o comportamento do chicote?
- Como está o comportamento do bumerangue?
- O chicote/bumerangue funciona normalmente quando a personagem está desempenhando outras ações (jogar bumerangue/chicote, pular, usar turbo)?
- Os chicotes das personagens funcionam normalmente ao serem lançados um contra o outro?
- O chicote apresenta algum comportamento estranho em algum lugar específico do cenário?

Integração Arte/Som/Jogabilidade

- As animações e os sons se relacionam bem?
- As ações e os sons se relacionam bem?
- As ações e as animações se relacionam bem?
- As músicas e a arte conceitual trabalham juntas? Complementam-se?
- As animações, sons e comandos estão sincronizados?

Diferentemente do que acontece nas tarefas de testes no desenvolvimento de um software convencional, o papel dos testadores não era simplesmente testar as funcionalidades do *game*, verificando se elas apresentavam problemas ou incoerência. Tão importante quando isso, o clima e a diversão eram analisados a fundo, para garantir que o *game* desempenhasse o papel que propunha.

É importante ser dito, contudo, que a equipe de *Game Testers* era constituída por indivíduos que possuíam conhecimentos técnicos em áreas específicas, como Arte e Código. Esses conhecimentos não abrangiam as tarefas específicas das equipes de desenvolvimento do *game* em questão, mas sim aspectos gerais. É importante fazer essa distinção, já que se os *Game Testers* estivessem a par dos produtos do desenvolvimento, poderiam ser influenciados na tarefa de testes.

Isso ajudou a dar um olhar também mais técnico dentro de um ambiente de testes "humano".

O papel dos testadores era fazer uma análise crítica à estória e à integração das partes de desenvolvimento ("criando" e dando vida a essa estória), observando em cada aspecto o que viria a melhorar ou a piorar se adicionado ou removido do produto.

Porém, com os conhecimentos que tinham, os testadores poderiam também ver aspectos específicos de determinada área e encontrar erros com mais facilidade, bem como entender o motivo dos eventuais problemas encontrados. Além disso, como eram pedidas as opiniões pessoais de cada *Game Tester*, as áreas tinham muito a ganhar com o relatório entregue no final dos testes.

4.5.4 Testes pela equipe de desenvolvimento como *Game Testers*

Até então, a diferenciação básica entre as duas equipes de teste se dava pelo fato de que, enquanto os desenvolvedores testavam se a integração entre as áreas do desenvolvimento funcionava, os *Game Testers* verificavam se essa integração era convincente. Nesta nova abordagem, contudo, foi proposto um novo tipo de teste.

Além dos testes feitos pelos *Game Testers* (analisando o *game* com um olhar crítico do próprio público alvo) e dos testes feitos pela equipe (verificando a consistência das integrações de todas as partes do desenvolvimento), foram também realizados testes de uma

forma diferente: Com uma visão voltada ao produto final, como a dos *Game Testers* (sem se preocupar em como foram implementadas e integradas suas funcionalidades) pela equipe de desenvolvimento.

Esse novo tipo de teste apresentou resultados positivos para o produto final, listados a seguir.

- **Erros e possibilidade de melhorias em áreas específicas mais facilmente encontrados**

Obviamente, ao testarem o *game*, os membros de cada equipe tendiam a analisar mais profundamente aspectos que diziam respeito às suas áreas específicas. Assim, enquanto os profissionais da equipe de Som se concentravam nos *loops* das músicas, por exemplo, os *Game Designers* prendiam suas atenções aos movimentos da personagem dentro do *game*. Isso fez com que fossem encontrados problemas técnicos até então não observados.

Embora muitas melhorias tenham sido feitas ao longo do processo de desenvolvimento baseando-se nos resultados dos testes de integração feitos pelas equipes, alguns aspectos falhos ou pouco trabalhados foram encontrados apenas no exercício do *game*. Isso se deu porque certos detalhes passavam desapercebidos no desenvolvimento, quando vistos separadamente, mas tornavam-se claros com todos os objetos unidos em cena, formando o mundo do *game* em si.

Experimentando aquele mundo, o desenvolvedor pôde ser capaz de ver e entender melhor determinados problemas e imperfeições, que só podiam ser vistos sob uma ótica mais técnica e que, por isso, passaram desapercebidos também pela equipe de *Game Testers*.

- **Erros em outras áreas mais facilmente encontrados**

É fácil constatar que quando um trabalho é executado por um longo período de tempo pelas mesmas pessoas responsáveis, as chances de serem encontradas inconsistências diminuem. Se a equipe não encontrou problemas em uma etapa do desenvolvimento, é possível que continue não encontrando até que ela termine, fazendo assim com que o erro se propague.

Por isso, é muito importante um olhar crítico de um profissional que esteja “de fora”. Como sua mente não sabe o que ele está procurando, este profissional não será guiado

exatamente para um ponto específico, mas olhará o produto como um todo e poderá analisá-lo de forma homogênea.

Ao jogar o *game*, houve situações em que profissionais de uma área foram capazes de detectar erros ou imperfeições em aspectos relacionados ao trabalho das outras equipes. Isso foi facilitado pelas discussões que as equipes mantiveram durante o desenvolvimento, nas *Game Design Weekly Meetings*.

Nessa reunião, eram discutidos aspectos importantes em cada área de desenvolvimento. Muitas vezes, membros de uma equipe acabavam explicando detalhes técnicos específicos de sua área para profissionais de outras equipes, de forma que a análise de novas idéias e sugestões fosse feita com base no maior número de informações disponíveis.

Assim, o fluxo de conhecimento entre as equipes gerou uma visão "não tão leiga, não tão técnica" sobre uma determinada área aos profissionais que estavam fora dela. Estes profissionais encontravam, eventualmente, correções a serem feitas em um determinado aspecto do jogo, as quais não puderam ser detectadas pelos olhos mais experientes dos membros da equipe responsável, já que estes procuravam outros tipos de inconsistências (julgadas mais importantes ou impactantes para o produto final).

▪ **Teste de qualidade para um público mais técnico**

Como já dito, os consumidores estão buscando cada vez mais qualidade nos *games* lançados. Essa busca resulta no aumento de um público altamente crítico e "perfeccionista", que avalia não somente o *game* como um todo, mas também seus mínimos detalhes. Uma prova disso é a grande quantidade de material disponibilizado na internet (vídeos, fotos e texto) por jogadores que mostram e detalham imperfeições e *bugs* encontrados em determinados *games*.

Com isso, é possível notar que os *games* estão reunindo um público mais técnico e mais focado nos detalhes do trabalho apresentado. Embora isso, de certa forma, ajude os desenvolvedores, informando-os sobre problemas encontrados, por outro lado denigre a imagem dos responsáveis pelo *game*.

Por isso, os testes feitos pelos desenvolvedores na ótica de *Game Testers* se mostrou de grande utilidade para melhorar o trabalho no ponto de vista técnico. Assim como o público

que critica seus trabalhos, os desenvolvedores têm a oportunidade de procurar erros e enxergar, antes dos jogadores, todos os aspectos técnicos do *game*.

Além dos resultados positivos obtidos para o produto em si, foram observados benefícios em relação ao trabalho das equipes de desenvolvimento. Primeiramente, as equipes puderam entender melhor como seus trabalhos "criavam" o jogo. Dessa forma, os desenvolvedores puderam analisar tanto os pontos técnicos como os gerais de suas próprias áreas e das áreas de outras equipes, fazendo com que fosse gerado um aprendizado natural ao longo da tarefa de teste.

Isso foi positivo porque, no projeto estudado, essa equipe desenvolvia um produto na versão *demo*, ou seja, o que seria apenas uma demonstração do que o *game* final viria a se tornar. Portanto, essa tarefa de teste deu aos membros das equipes a experiência do *game* em que estavam trabalhando, proporcionando a eles uma análise de como certas propriedades deveriam ser melhoradas ou modificadas para que o produto, em sua versão final, também melhorasse.

Além disso, testar o *game* em suas versões semi finais deu à equipe de desenvolvedores uma forma de distração e entretenimento, o que ajudou, em algumas ocasiões, a manter o equilíbrio entre o constante trabalho e o descanso, ajudando na própria produtividade da equipe. É interessante notar que a tarefa de testes, aqui, se mostrou (não inteiramente) oposta à tarefa de testes realizada pelos *Game Testers*: Enquanto para estes, jogar era o trabalho, para a equipe de desenvolvimento foi, em parte, uma forma de "dispersão".

Finalmente, os desenvolvedores puderam ganhar experiência da "ótica do cliente" para projetos futuros. Não apenas focados em encontrar erros no código ou na integração entre as áreas, os membros das equipes puderam também testar a diversão do seu próprio *game*, unindo as duas visões, técnica e "humana", no desenvolvimento de um produto.

4.6 Considerações Finais

Como visto, em se tratando de desenvolvimento de *games* (ou projetos que tenham mudanças freqüentes de requisitos em geral), é possível notar a vantagem em adotar uma metodologia ágil dentro da equipe de desenvolvedores, por ser facilmente adaptável a

mudanças. Contudo, foi observada a necessidade de adaptação de uma metodologia ágil (*Scrum*, neste trabalho) para a aplicação no ambiente de desenvolvimento de um *game*.

Essa adaptação foi de grande ajuda no cenário de testes do *game*. Desta forma, visões e estratégias de teste puderam ser aprimoradas e executadas de forma a ampliar o universo de testes, englobando visões específicas e "mistas" entre *Game Testers* e desenvolvedores, na busca de uma maior qualidade para o *game*.

Assim, com esta abordagem de testes, foi possível agregar valor não só ao produto, como também ao projeto e às equipes de desenvolvimento. Aprendendo a trabalhar com mais ênfase em equipe, esses profissionais puderam interagir entre si, ganhando experiência profissional e desempenhando também o papel de testadores do *game*, resultando no aumento de qualidade do produto final.

5. Conclusão

5.1 Contribuições

Como *games* apresentam aspectos que devem ser levados em consideração no momento de testes, este trabalho se propôs a mostrar uma maior abrangência e uma nova visão na tarefa de testes dentro do desenvolvimento de um *game*. Foram abordadas, assim, maneiras específicas de avaliar a jogabilidade e aspectos técnicos, tentando diminuir os desafios encontrados entre as tarefas de testes de um software e o desenvolvimento de *games*.

Assim, foi reafirmada a idéia da importância de testes na integração entre as equipes do desenvolvimento do *game*. Obviamente, em qualquer software produzido, as partes devem estar bem integradas, de forma que resultem em um produto de qualidade. É importante afirmar, porém, que essa integração no desenvolvimento de um *game*, especificamente, tem um grau ainda maior de importância para o produto final, já que este produto não apresenta apenas aspectos técnicos, mas também subjetivos. Enquanto um sistema de informação, por exemplo, se propõe a apenas funcionar corretamente, sem apresentar *bugs*, um *game*, além disso, deve ter elementos emocionais e específicos da natureza humana para ser considerado de qualidade.

Além dos testes de integração entre as partes pela equipe de desenvolvimento, foi observada a vantagem de ter o *game* testado por *Game Testers* com conhecimentos específicos, diferentemente de um jogador. Independentemente da quantidade de testes feitos no ato de jogar o *game*, um olhar mais técnico certamente faz com que o testador possa encontrar problemas mais específicos relacionados aos seus conhecimentos. Isso acontece principalmente porque estes testadores sabem onde e o que devem procurar. Com mais experiência do que um usuário, eles podem detectar, com mais facilidade, aspectos que podem gerar conflitos no andamento do *game*, forçando ocasiões em que esses aspectos sejam mostrados e, eventualmente, falhem.

Como dito, essa abordagem faz com que seja dado um olhar mais técnico a uma parte de testes predominantemente “humana”, mesclando de forma eficaz dois pontos importantes a serem considerados no produto final: A funcionalidade e a diversão do *game.*

Por fim, foram identificadas as vantagens em se adotar testes que sejam executados por testadores com visões técnicas sobre uma ótica mais subjetiva. Neste caso, os desenvolvedores do *game* em questão desempenharam o papel de *Game Testers*, verificando o clima do jogo e a interação entre os elementos resultantes de seus próprios trabalhos.

Com isso, erros que passaram desapercebidos puderam ser encontrados com mais facilidade, tanto pela equipe responsável pela parte que apresentava o erro (por buscarem com mais foco problemas específicos) quanto pelas outras equipes (por testarem a integração das partes do *game* e possuírem um melhor conhecimento para fazê-lo, proveniente das *Weekly Game Design Meetings*).

Isso não só aumentou a qualidade do *game*, por diminuir a quantidade de erros não identificados, como também ajudou a própria equipe de desenvolvimento a atuar de uma forma mais abrangente e mais eficaz, identificando e entendendo com mais facilidade problemas de integração que aconteceram e que poderiam vir a acontecer com novas funcionalidades adicionadas ao *game* em sua versão final, bem como conflitos entre as partes em projetos futuros. É fácil notar, portanto, que os desenvolvedores amadureceram e se especializaram no trabalho em equipe, o que é primordial para o resultado de um *game* de qualidade.

5.2 Limitações

Durante o desenvolvimento do game estudado nesta monografia, houve uma alta rotatividade dos membros nas equipes, principalmente na equipe de Código. Com as freqüentes mudanças na equipe, não foi possível fazer um levantamento com os profissionais envolvidos sobre suas opiniões em relação à abordagem descrita neste trabalho, perdendo-se assim um pouco do valor real do resultado desta proposta.

Além disso, essa alta rotatividade limitou uma integração mais ampla entre as equipes do desenvolvimento. Como essa integração não ocorreu em todo o tempo do desenvolvimento com os mesmos profissionais, a disseminação dos conhecimentos técnicos entre os membros das diferentes equipes foi afetada.

O valor do resultado obtido neste trabalho seria também mais significativo se esta abordagem fosse adotada por uma equipe maior ou por mais equipes de desenvolvimento de um *game*. Infelizmente, só foi possível analisar de perto o trabalho de uma única equipe.

5.3 Trabalhos Futuros

Como possível trabalho futuro, seria interessante um estudo que propusesse um aprofundamento nas técnicas de testes para *games* que sejam desenvolvidos com uma metodologia tradicional de gerenciamento de projeto, diferentemente do *Scrum*, bem como uma adaptação da metodologia estudada à esta área. Como métodos ágeis se encaixam de uma maneira mais eficiente no desenvolvimento de um *game*, como já discutido neste trabalho, seria interessante uma proposta de testes para a área de *games* dentro de um desenvolvimento direcionado por métodos tradicionais, buscando compreender, nesta visão, as maiores dificuldades entre estas duas áreas – *games* e testes de software – bem como diminuir esses desafios.

Além disso, seria interessante a comparação entre os testes de software para *games* dentro de um ambiente com métodos tradicionais e um ambiente com métodos ágeis, indicando e explicando as vantagens e desvantagens entre eles e propondo novas formas de torná-los igualmente ou aproximadamente eficazes.

6. Referências Bibliográficas

AMMANN, P. OFFUTT, J. *Introduction to software testing*. Estados Unidos: Cambridge University Press, 2008. 322 p.

BARTIE, A. *Execução dos serviços de testes*. Disponível em: <http://imasters.uol.com.br/artigo/4991/des_de_software/execucao_dos_servicos_de_testes/> Acesso em: 2009.

BARTIE, A. *Gerenciamento dos serviços de testes*. Disponível em: <http://imasters.uol.com.br/artigo/4948/des_de_software/gerenciamento_dos_servicos_de_testes/> Acesso em: 2009.

BARTIE, A. *Inovação dos serviços de testes*. Disponível em: <http://imasters.uol.com.br/artigo/5109/des_de_software/inovacao_dos_servicos_de_testes/> Acesso em: 2009.

BARTIE, A. *Processo de teste de software*. Disponível em: <http://imasters.uol.com.br/artigo/6102/des_de_software/processo_de_teste_de_software_-_parte_01/> Acesso em: 2009.

BERGAMI, Paul. *Qualidade – prevenção de falhas.* Disponível em: <http://paulbergami.blogspot.com/2009/07/defeitos-sao-caracterizados-como.html > Acesso em: 2009.

BETHKE, E. *Game development and production.* Texas: Wordware Publishing, 2003. 412 p.

BLIZZARD ENTERTAINMENT. *World of Warcraft.* Disponível em: <http://www.worldofwarcraft.com> Acesso em: 2009.

COLLINS-SUSSMAN, B. FITZPATRIC, B. PILATO, C. *Version control with Subversion.* O'Reilly Media, 2008. 432 p.

DIAS, Sérgio Almeida. *Qualidade de software e testes.* Disponível em: <http://www.sergiodias.inf.br/engenharia-de-software/qualidade> Acesso em: 2009.

FELICIANO, Fátima Aparecida. *Indústria do entretenimento na era da convergência digital.* Disponível em: <http://www.rp-bahia.com.br/biblioteca/hist-midia2005/resumos/R0020-1.pdf> Acesso em: 2009.

FOLHA ONLINE. *Pesquisadores comparam vício no jogo "World of Warcraft" ao uso de crack.* Disponível em: <http://www1.folha.uol.com.br/folha/informatica/ult124u510481.shtml> Acesso em: 2010.

GARAGE GAMES. *Torque X 2D.* Disponível em: < http://www.torquepowered.com/products/torquex-2d > Acesso em: 2010.

JOOSE. *What are AAA games?* Disponível em: < http://www.gameproducer.net/2006/05/26/what-are-aaa-titles/ > Acesso em: 2010.

MACORATTI, José Carlos. *Um esboço sobre o processo de testes de software*. Disponível em: < http://www.macoratti.net/08/08/tst_sw2.htm> Acesso em: 2009.

MICROSOFT CORPORATION. *Microsoft Visual Studio*. Disponível em: < http://www.microsoft.com/express/Downloads/#2008-Visual-CS > Acesso em: 2010.

MICROSOFT CORPORATION. *XNA developer center.* Disponível em: < http://msdn.microsoft.com/en-us/aa937791.aspx > Acesso em: 2010.

MULLER, A.drian Maciel Laubisch. *Scrum4Games – Uma adaptação da metodologia Scrum para o desenvolvimento de projetos de games.* Trabalho de conclusão de curso (Bacharelado em Ciência da Computação). Niterói, 2009. 51 p.

MYERS, Glenford J. *The art of software testing*. Nova Jersey: John Wiley & Sons, 2004. 234 p.

NAIK, K. TRIPATHY, P. *Software testing and quality assurance: theory and practice*. Nova Jersey: John Wiley & Sons, 2008.

NOGUEIRA, Elias. *Palestra ALATS SP - FIAP teste de software.* Disponível em: <http://www.slideshare.net/elias.nogueira/palestra-alats-sp-fiap-teste-de-software> Acesso em: 2009.

PALMA, Fernando. *O custo da não qualidade.* Disponível em: <http://testesdesoftware.blogspot.com/> Acesso em: 2009.

PC WORLD. *Indústria de games movimenta US$ 9,5 bilhões durante 2007, diz NPD.* Disponível em: <http://pcworld.uol.com.br/noticias/2008/01/25> Acesso em: 2009

PILONE, D. MILES, R. *Head first software development*. Estados Unidos: O'Reilly Media, 2007. 447 p.

PRESSMAN, R. *Engenharia de software.* 6 ed. São Paulo: McGraw-Hill Interamericana do Brasil, 2006. 720 p.

RÉGIS, Alex. *Vídeo Game: risco, vício ou prazer?* Disponível em: <http://tribunadonorte.com.br/noticia.php?id=97752 > Acesso em: 2009.

ROUSE, R. *Game design: theory & practice*. Texas: Wordware Publishing, 2001. 584 p

SCHWABER, K. BEEDLE, M. *Agile Software Development with Scrum*. Prentice Hall, 2001. 158 p.

TOZELLI, P. *Teste de software*. Disponível em: <http://imasters.uol.com.br/artigo/9572/des_de_software/teste_de_software/> Acesso em: 2009.

TOZELLI, P. *Análise do teste de software*. Disponível em: <http://imasters.uol.com.br/artigo/9817/des_de_software/analise_do_teste_de_software_-_parte_02/> Acesso em: 2009.

VIDEOGAMETESTERHUB.COM. *Game tester tips.* Disponível em: <http://www.videogametesterhub.com/game-testing-tips.html#more-185> Acesso em: 2009.

VIDEOGAMETESTERHUB.COM. *What does a vídeo game tester do?* Disponível em: <http://www.videogametesterhub.com/what-does-a-video-game-tester-do.html#more-281> Acesso em: 2009.