NON-FUNCTIONAL REQUIREMENTS FRAMEWORK: INFLUÊNCIAS NA QUALIDADE DE JOGOS DE ENTRETENIMENTO COM TEMÁTICA *ROLE-PLAYING GAME*

Gustavo Garcia dos Reis Nunes[1];André Castro Leal[2]

[1] Discente do curso de Sistemas de Informação da Universidade Federal Rural do Rio de Janeiro, [2]Docente do Departamento de Computação/ICE/UFRRJ
Seropédica – RJ – Brazil

gustavogarcia@pet-si.ufrrj.br

RESUMO

1. INTRODUÇÃO

Desenvolver *software* é uma operação complexa por natureza, uma das razões para esta afirmação é que não existe apenas uma única resolução para cada cenário de desenvolvimento (PRESSMAN, 2016). Por isso, temos um estudo na área da computação voltada para entender melhor essa complexidade, a Engenharia de *Software (ES)*. Lidamos o tempo todo com pessoas, o que torna o sucesso do projeto bastante relacionado à competência da equipe e à forma como trabalham, e, para dificultar ainda mais, muitas vezes não fazemos uso de um processo bem definido para apoiar as atividades do projeto.

Com isso, uma das áreas do conhecimento da Engenharia de *Software,* é a Engenharia de Requisito. Ela é definida como as partes interessadas de um novo sistema, que é baseada em objetivos do mundo real que possui o propósito de satisfazer as necessidades do sistema (PRESSMAN,2016). Os requisitos guiam as atividades do projeto e normalmente são expressos em linguagem natural para que todos possam obter o entendimento. Podemos definir que a engenharia de requisitos é um processo que engloba todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo.

Portanto, a engenharia de requisitos é o processo pelo qual os requisitos de um produto de software são coletados, analisados, documentados e gerenciados ao longo de todo o ciclo de vida do software.

Podemos relacionar esse tipo de processos com a Engenharia de Requisitos Orientada a Objetivos (do inglês, *Goal-Oriented Requirements Enginering* ou *GORE*). Este método tem crescido como uma forma promissora de como descrever sistemas de *software*. Ela baseia-se nos objetivos dos *stakeholders*, com a finalidade de desenvolver o *software* que realmente satisfaça os desejos deles. Portanto, entende-se que é uma exigência ou estado do mundo que um *stakeholder* gostaria de alcançar.

Então, podemos associar que o desenvolvimento de *Software* não é diferente de desenvolvimento de jogos de entretenimento, ambos tratam de um projeto usando artefatos computacionais, tendo um escopo, prazo e qualidade. O processo de desenvolvimento de jogos eletrônicos, ou *games*, costuma ser bastante caótico e complicado. A inconstância de diversos aspectos inerentes ao desenvolvimento, como os requisitos a se satisfazer e as tecnologias disponíveis, faz com que a adoção de processos rıgidos e bem definidos seja

bastante dificultada, muitas vezes levando equipes inteiras a seguir processos “artesanais” ou “artísticos”.

Desta forma, são traçados paralelos diversos entre as fases do desenvolvimento, assim como os processos e técnicas adotados nelas, como as metodologias clássicas da Engenharia de *Software*. Todos os passos do desenvolvimento são apresentados de forma simplificada, trazendo uma visão genérica de como funcionam os processos mais utilizados.

Incorporado no desenvolvimento de *games,* temos uma temática denominada *role playing games (RPG),* em uma adaptação livre, significa “jogos de interpretação de papéis”. O produto foi elaborado nos Estados Unidos, no início da década de 70, pelos estudantes de história Gary Gygax e Dave Arneson, e inaugurado pelo jogo de tabuleiro *Dungeons & Dragons* (Catacumbas & Dragões). Em suas primeiras versões,trata de um jogo que contém um conjunto de regras definidas em um universo medieval fictício, inspirado nas velhas histórias de cavaleiros andantes, e um grupo de personagens, a serem interpretados pelos jogadores.

No mundo dos jogos de entretenimento, essa temática é mundialmente conhecida, com os jogos consolidados pelo mercado e aclamado pela crítica, alguns exemplos de *games é* a série *Final Fantasy e Dragon Quest,* ambos desenvolvidos pelo estúdio japonês *square-enix.*

Portanto, objetivo deste trabalho é usar a modelagem *Non-Functional Requirements framework (NFR)* que consiste em uma abordagem orientada a processos, onde os requisitos não-funcionais são explicitamente representados como metas a serem obtidas. Estão relacionados com o comportamento dum sistema e não com as suas funcionalidades – descrevem como o sistema faz e não o que faz, são normalmente, representados por gráficos *Softgoal Interdependency Graph* (*SIG*), que descrevem as dependências entre os *softgoals* e como eles são decompostos. Em vista disso, este estudo vai discutir a representatividade que um modelo *NFR* tem na qualidade em um jogo eletrônico de entretenimento com tema *RPG* baseando se em resultados obtidos em trabalhos já realizados, e também comparar como essa ferramenta pode afetar a qualidade do produto.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Engenharia de *Software* Tradicional

Um processo de software (ou metodologia de desenvolvimento de software) é um conjunto de atividades e resultados associados que auxiliam na produção de software. Dentre as várias atividades associadas, existem por exemplo a análise de requisitos e a codificação. O resultado do processo é um produto que reflete a forma como o processo foi conduzido.

Existem diversos processos de *software* definidos na literatura da Engenharia de *Software*. É frequente que em algumas organizações criarem seu próprio processo ou adaptar algum processo à sua realidade. Dentre os vários processos existentes, existem as metodologias tradicionais, que são orientadas a documentação.

As metodologias tradicionais são também chamadas de pesadas ou orientadas a documentação. Essas metodologias surgiram em um contexto de desenvolvimento de software muito diferente do atual, baseado apenas em um mainframe e terminais burros (Royce,1970). Na época, o custo de fazer alterações e correções era muito alto, uma vez que o acesso aos

computadores eram limitados e não existiam modernas ferramentas de apoio ao desenvolvimento do software, como depuradores e analisadores de código. Por isso o software era todo planejado e documentado antes de ser implementado. A principal metodologia tradicional e muito utilizada até hoje é o modelo Cascata (Figura 1).

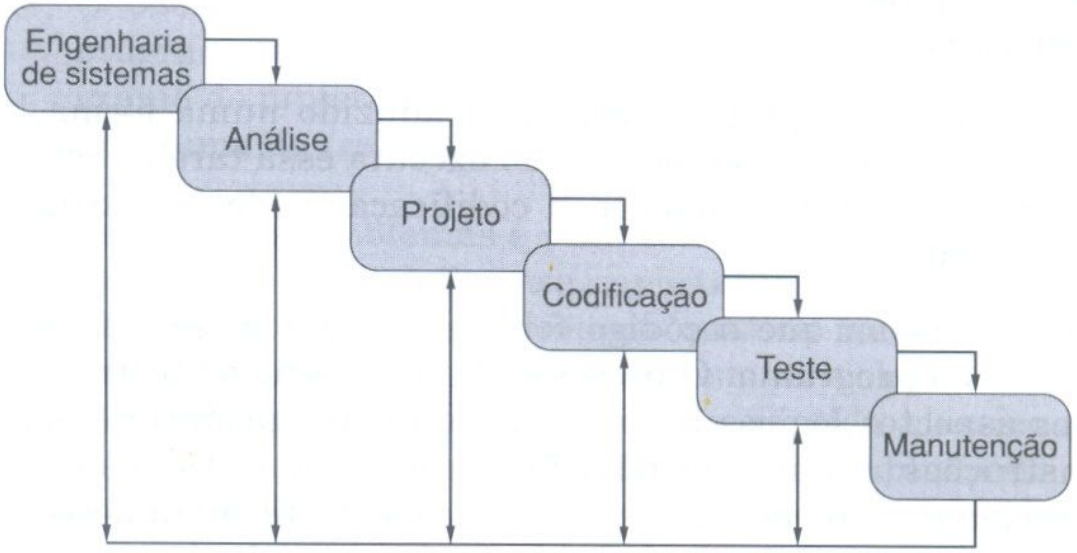

Figura 1: Modelo de desenvolvimento de *Software,* Cascata

O modelo Cascata ou Clássico (Pressman, 2016) foi o primeiro processo publicado de desenvolvimento de *software*. Desde sua introdução tem sido muito utilizado, ele é um modelo em que existe uma sequência a ser seguida de uma etapa a outra. Cada etapa tem associada ao seu término uma documentação padrão que deve ser aprovada para que se inicie a etapa imediatamente posterior. De uma forma geral fazem parte do modelo Clássico as etapas de definição de requisitos, projeto do *software*, implementação e teste unitário, integração e teste do sistema, operação e manutenção.

A abordagem adotada pela metodologia cascata acaba trazendo alguns problemas. Dentre estes problemas merece destaque o fato de que os projetos reais dificilmente seguem o fluxo sequencial, o cliente quase sempre não consegue exprimir todas as suas necessidades além de ser exigida dele muita paciência visto que o software só estará pronto para uso num ponto tardio do cronograma. E o maior dos problemas é que se ocorrer um erro em qualquer uma das etapas o resultado pode ser desastroso e frequentemente caro (PRESSMAN, 2016).

O modelo Clássico dominou a forma de desenvolvimento de software até o início da década de 90, apesar das advertências dos pesquisadores da área e dos desenvolvedores, que identificaram os problemas gerados ao se adotar esta visão seqüencial de tarefas. Por exemplo, Fred Brooks em seu artigo *"No Silver Bullet: Essence and Accidents of Software Engineering"*, descreve que a idéia de especificar totalmente um *software* antes do início de sua implementação é impossível (BROOKS,1987). Outro pesquisador, Tom Gilb, desencoraja o uso do modelo Clássico em grandes *softwares*, estimulando o desenvolvimento incremental como um modelo que apresenta menores riscos e maiores possibilidades de sucesso (GILB,1999).

Algumas vezes o cliente define um conjunto de objetivos gerais que não esclarecem consistentemente os requisitos. Outras vezes o desenvolvedor não tem certeza da eficiência de parte do código, da adaptação da aplicação ao sistema operacional ou mesmo da forma que interação homem-máquina deve ter. Visando fornecer melhores soluções à casos assim surgiu a metodologia conhecida como prototipação.

A Prototipação (Figura 2) é uma metodologia surgida posteriormente à Cascata. Ela possibilita a equipe de desenvolvimento a criar uma aplicação protótipo que pode assumir três formas distintas. A primeira delas é um protótipo em papel ou mesmo no computador que retrata a interação homem-máquina. A segunda opção é implementar uma funcionalidade que já está no escopo do *software* a ser desenvolvido. Por fim, existe a possibilidade de utilizar-se de um *software* já pronto que tenha parte ou todas as funcionalidades desejadas. Esta forma é mais comumente adotada em softwares que apesar de prontos ou parcialmente prontos possuem características que precisam ser incrementadas ou melhoradas em um novo esforço de desenvolvimento (PRESSMAN, 2016).

Figura 2: Representação da metodologia de prototipação

A Prototipação também possui pontos negativos. Um deles, é que o cliente pode acreditar que o protótipo já é o *software* pronto ou em fase de término e começar pressionar para que se faça pequenos ajustes e entregue o *software* rapidamente. Diante de um quadro assim, muitas vezes, a qualidade final perde importância e acaba finalizando em um sistema inacabado com erros e mal implementado, e com isso, a manutenibilidade pode ficar comprometida. Outro ponto negativo a ser considerado, é que algumas vezes a equipe de desenvolvimento pode fazer decisões precipitadas com o intuito de colocar o protótipo em funcionamento, resultando, na maioria dos casos, no produto final.

Apesar desses problemas, a prototipação ainda é uma eficiente metodologia de desenvolvimento de *software*. A fim de se obter sucesso no projeto, tanto cliente quanto desenvolvedor devem chegar a um consenso de que o protótipo servirá apenas para ajudar na

definição dos requisitos (PRESSMAN, 2016). Apesar de resolverem muitos dos problemas do desenvolvimento de *software* alguns parâmetros ainda não eram fornecidos pelas metodologias existentes.

Por fim, é válido destacar a metodologia espiral (figura 3), que foi concebida para englobar as melhores práticas tanto do ciclo de vida clássico quanto da prototipação. Essa metodologia inovou ao trazer também um novo elemento, a análise de riscos. Além disso, foi uma das primeiras metodologias a adotar o conceito de iteração. Sucessivas iterações moldam aos poucos soluções mais completas do software (PRESSMAN, 2016).

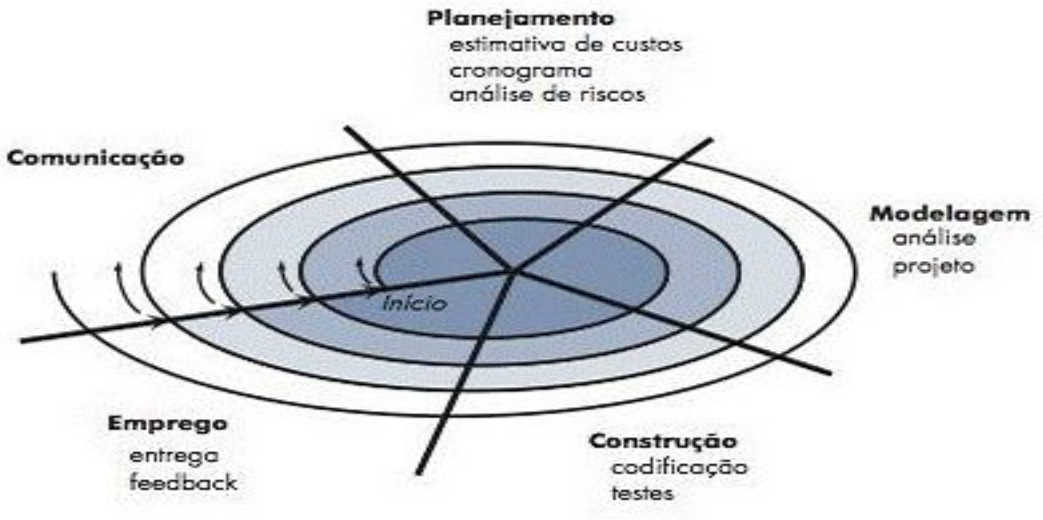

Figura 3: Metodologia Espiral de desenvolvimento de *software.*

Na primeira iteração, os objetivos, alternativas e restrições são definidos e os riscos são identificados e analisados. O cliente avalia o resultado da iteração e baseado nos apontamentos do mesmo a próxima iteração é iniciada. Isso possibilita ao cliente e ao desenvolvedor perceber e reagir a riscos em cada uma das etapas evolutivas. Entretanto a metodologia espiral exige considerável experiência para avaliar os riscos e se baseia nela para obter sucesso.É visto que se um grande risco não for detectado certamente ocorrerão problemas.

2.2 Engenharia de Requisitos Orientada a Objetivos

A pesquisa tradicional de engenharia de requisitos começa com as declarações dos primeiros requisitos, que expressam os desejos do cliente sobre "o que" o sistema deve fazer. O objetivo das tarefas tradicionais de engenharia de requisitos é produzir um documento de requisitos para transmitir aos desenvolvedores. O objetivo é que o sistema resultante seja adequadamente especificado e restrito, muitas vezes em um ambiente contratual. Ele ignora o foco em “por que” o sistema deve fazer, que é o foco do *GORE*.

Como já discutido na seção anterior, existem diversas abordagens para a modelagem e especificação de de um produto de *software*. É essencial apresentar para esse estudo,um processo que compõe o desenvolvimento de *software,* a Engenharia de Requisitos.Uma das muitas abordagens é a orientada a objetivos. Modelos orientados a objetivos mostraram-se efetivos para identificar variabilidade nas fases iniciais de requisitos, permitindo a captura de

formas alternativas para alcançar os objetivos dos *stakeholders* (LIASKOS, LAPOUCHNIAN, et al., 2006).

A Engenharia de Requisitos Orientada a Objetivos visa preencher as lacunas existentes no desenvolvimento de software, baseando-se nos objetivos dos *stakeholders*, de modo que o *software* a ser desenvolvido corresponda ao que realmente os *stakeholders* desejam (LAMSWEERDE, 2001). Uma abordagem *GORE* particular, que tem sido foco de muitos trabalhos nos últimos anos, é o *framework i** (YU, 1995). Neste *framework*, o ambiente de operação do sistema e o sistema em si são vistos como atores organizacionais, que dependem da ajuda uns dos outros para que seus objetivos sejam cumpridos. O i* permite a construção de duas visões de modelos.

2.3 *Non-functional requirements framework*

Basicamente, a utilidade de um sistema é definida por sua funcionalidade funcional e não funcional. Características,como usabilidade, flexibilidade, desempenho, interoperabilidade e segurança. Além disso, o conceito de qualidade também é fundamental para a Engenharia de *Software* e as características funcionais e não funcionais devem ser levados em consideração no desenvolvimento de um sistema de *software* de qualidade.

Contudo, devido à curta história por trás da Engenharia de *Software*, em parte devido à demanda em ser cada vez mais veloz, os sistemas em execução que atendam a necessidade básica, e também em parte devido à natureza "suave" dos requisitos não funcionais, a maior parte da atenção do desenvolvimento de sistemas no passado tem sido centrada em notações e técnicas para definir e fornecendo as funções que um sistema de *software* deve executar.

A engenharia de requisitos vem classificando requisitos como funcional ou não funcional, a maioria dos modelos e requisitos de requisitos existentes as linguagens de especificação não tinham um tratamento adequado das características de qualidade.O tratamento das características de qualidade como um todo e não apenas como funcionalidade tem sido um dos principais focos de trabalhos na área de engenharia de requisitos orientada por objetivos (LAMSWEERDE, 2001), e em particular o *NFR Framework* (CHUNG,1999), que trata a não funcionalidade em alta nível de abstração para o problema e a solução.

Non-Functional Requirements é um *framework* para modelagem de metas (Figura 4). A análise começa com *softgoals* que representam o *NFR* com o qual as partes interessadas concordam. *Softgoals* são objetivos difíceis de expressar, mas tendem a ser qualidades globais de um sistema de *software*. Estes podem ser usabilidade, desempenho, segurança e flexibilidade em um determinado sistema. Esses *softgoals* são então geralmente decompostos e refinados para descobrir uma estrutura de árvore de metas e sub-objetivos por exemplo. Uma vez que as estruturas de árvores são descobertas, uma é obrigada a encontrar metas flexíveis interferentes em árvores diferentes, por ex. os objetivos de segurança geralmente interferem na usabilidade. Essas árvores flexíveis agora formam uma estrutura gráfica flexível. O passo final nesta análise é escolher algumas metas flexíveis específicas das folhas, de modo que todas as metas flexíveis da raiz sejam satisfeitas.

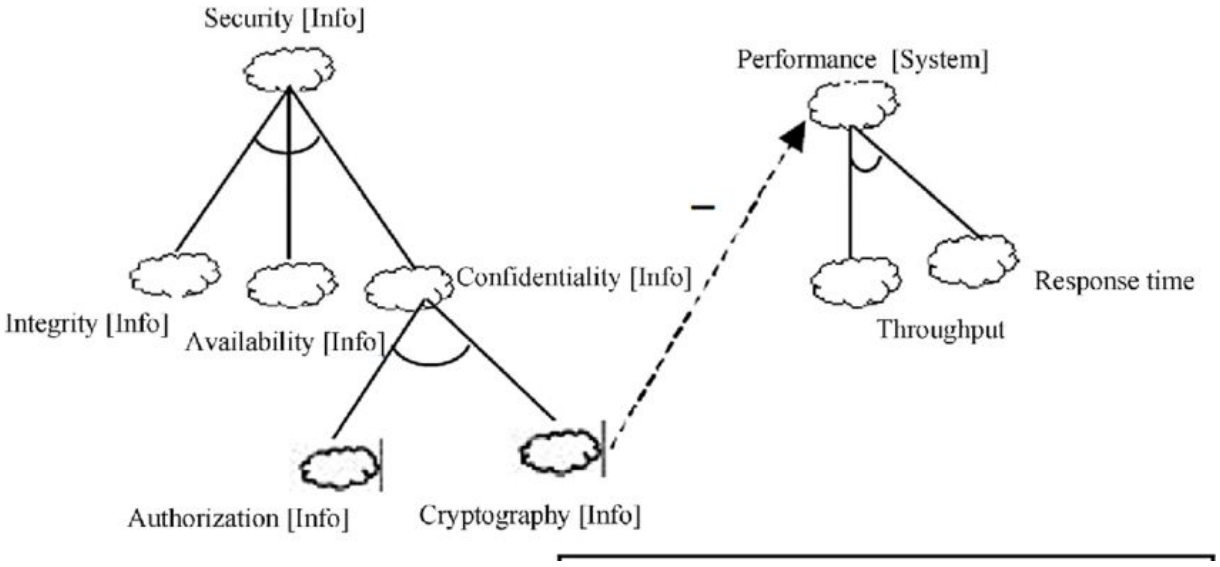

Figura 4: Modelo de um *NFR*.

2.4 Jogos Eletrônicos de Entretenimento

Um jogo eletrônico, teve início quando os acadêmicos começaram a projetar jogos simples, simuladores e programas de inteligência artificial, como parte de suas pesquisas em ciência da computação. Somente a partir das décadas de 1970 e 1980 é que os jogos eletrônicos se tornaram populares, quando jogos de *arcade*, console de jogos eletrônicos e jogos de computador foram introduzidos ao público em geral. Desde então, os jogos eletrônicos tornaram-se uma forma popular de entretenimento e uma parte da cultura moderna em diversas regiões do mundo (LEITE, 2003).

Atualmente, já passamos por 9 gerações de consoles eletrônicos, *arcades,* videogames portáteis e periféricos de realidade aumentada. Empresas do mundo todo trabalham nesta área do entretenimento, cada geração possui uma vasta biblioteca de jogos dos mais variados gêneros, tendo títulos como *Grand Theft Auto V,* a atingir a marca de 6 Bilhões de copias vendidas, sendo o produto de entretenimento mais rentável já produzido(*MarketWatch,* 2018).

Os *games* eletrônicos estão ganhando cada vez mais espaço em todo o mundo e já são considerados a nova revolução da mídia.Uma prova disso é que, nos últimos anos, o setor de games cresceu mais do que o cinema e a televisão (STAMFORD, 2013). O mercado brasileiro de jogos eletrônicos também está se tornando cada vez mais atrativo, chegando a movimentar em torno de US$1,3 bilhões por ano (NEWZOO, 2017).

2.5 Jogos Eletrônicos de Entretenimento com temática *RPG*

Para complementar as informações neste estudo, é necessário destacar um gênero de entretenimento bastante conhecido no mercado, o *Rolling Playing Game (RPG)*. Essa temática, consiste em uma situação onde o jogador controla as ações de um personagem imerso num

mundo definido, incorporando elementos dos *RPGs* tradicionais, compartilhando geralmente a mesma terminologia, ambientações e mecânicas de jogo. Outras similaridades com os *RPGs* de mesa incluem a ampla progressão de história e elementos narrativos, o desenvolvimento dos personagens do jogador, além de complexibilidade e imersão. Não existe um consenso claro sobre a definição do escopo exato do termo, especificamente variando se o foco na jogabilidade ou na história deve ser o elemento definidor. (VICENTE, 2016)

As características que define um *RPG* eletrônico são em geral inspirados em clássicos como *Dungeons & Dragons*, utilizando as mesmas terminologias, localizações e mecânicas de jogo. Os personagens possuem status, que são, de um modo geral, o HP (*Health Points* - Pontos de Vida), o MP (*Magic/Mana Points* - Pontos de Magia), Ataque, Defesa, Agilidade e Inteligência. Nos *RPG's* existe o sistema de níveis, no qual a cada batalha vencida, ou missão terminada, se recebe experiência, e uma certa quantidade de experiência leva o personagem ao próximo nível: o personagem fica mais forte, pode aprender magias novas, pode equipar novos tipos de armadura, etc. Geralmente existe um personagem principal para o jogo e um grupo que o acompanha. A história típica envolve um grupo de heróis obrigado a unir forças para cumprir uma missão, passando por diversos desafios pelo caminho.

3. MÉTODOS

3.1 Objetivos

O objetivo deste trabalho, é usar as técnicas da modelagem *NFR* representa na qualidade em um jogo eletrônico de entretenimento com tema *RPG,* usando resultados obtidos em trabalhos já realizados, como o da Maria Eleni Paschali, *Non-functional requirements that influence gaming experience: A survey on gamers satisfaction factors* e verificar como essa ferramenta pode afetar a qualidade do produto.

4. MODELO DE QUALIDADE

Para definir um modelo de qualidade, antes de tudo, vamos considerar a as experiências dos jogadores em relação aos *games* de *RPG,* tomando em base as mesmas decisões do estudo sendo utilizado como base, os subgêneros do *RPG* estão sendo agrupados ao gênero principal, para determinar um parâmetro de qualidade dos requisitos não funcionais, considerando que mesmo que possuem finalidades um pouco divergente, mas com o mesmo propósito de transmitir a temática *Rolling Playing Gaming* para os jogos eletrônicos.

A partir de uma pesquisa de satisfação realizada usando Escala Likert, o estudo em questão obteve os seguintes resultados em relações aos requisitos não funcionais(Tabela 1). Para determinar a qualidade do requisito, foi usado como parâmetro as seguintes escalas, da tradução livre, seriam eles do menos para o mais importante: Não importante, Relativamente não importante, neutro, importante e muito importante.

Os requisitos não funcionais levados em questão, foram: Cenário, Controles, Gráficos, Som,Velocidade de jogo, Comunidade do Jogo e Solidez dos Personagens.Cenário é um requisito voltado para a construção de um ambiente onde o jogador poderá explorar, seja ele uma floresta,cidade,caverna e entre outros. Controles são as mecânicas que influenciam os comandos do jogo, pode ser andar,correr,pular,atacar,pausar e entre outras funcionalidades. Gráficos é a parte de modelagem de texturas, da criação do visual do jogo. Som é toda parte sonora do jogo, desde vozes dos personagens, trilhas sonoras e sons ambientes. Velocidade do

jogo está ligado a performance do jogo, sendo ela de de tempo de carregamento de uma tela para outra até tempo de resposta de uma ação interagindo com a outra ação. o último requisito é a importância dos personagens jogáveis na trama e nas técnicas desenvolvidas no jogo.

Scale	Scenario	Controls	Graphics	Sound	Game Speed	Game Community	Character Solidness
Role Playing Games							
Not Important	1	1	0	0	2	2	1
Relatively Not Important	1	0	0	1	1	0	0
Neutral	6	10	8	10	7	6	11
Important	12	8	15	12	14	10	11
Very Important	8	8	4	5	4	10	5
Skewness	-0,99	-0,59	0,16	0,02	-1,07	-1,21	-0,57

Tabela 1: Resultados obtidos pela pesquisa de satisfação em relação aos jogos eletrônicos com temática *RPG*.

Para determinar qual o requisito mais importante a ser usado como modelo, os dados que receberam mais votos em importante e muito importante vão ser considerados como fator de qualidade do produto. Com isso, vamos destacar os 3 requisitos não funcionais que influenciam na satisfação dos jogadores de *RPG* eletrônico,, que são: Cenário, Velocidade de Jogo e Comunidade do Jogo. É válido salientar a notável relação dos fatores de satisfação com as descrições ditas sobre o conceito de *RPG* na seção 2.5. O único fator que desvirtua da descrição é a grande importância da Comunidade do Jogo. (Figura 5)

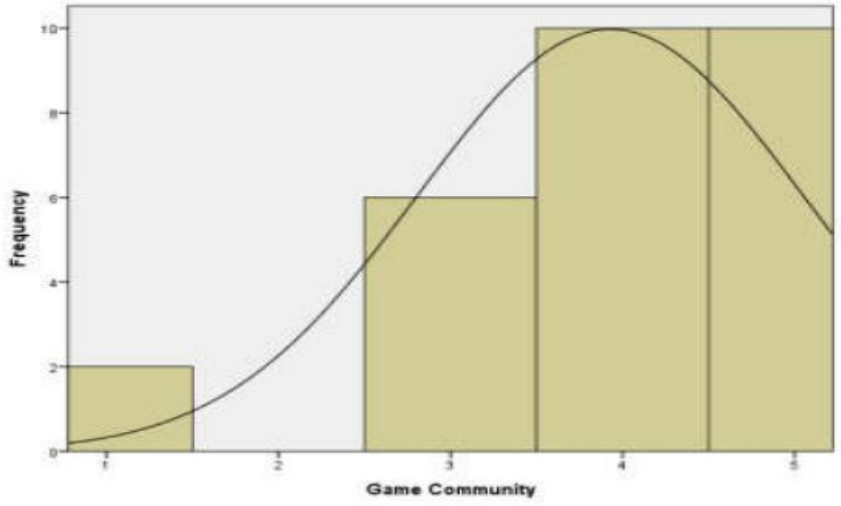

Figura 5: Gráfico de frequência dos dados obtidos sobre a Comunidade do Jogo.

5. EXEMPLOS DE MODELO DE QUALIDADE

A partir de uma análise e modelagem feita em *Non-functional Framework,* tivemos os seguintes exemplos de modelos de qualidade, um para cada requisito não funcional destacado.

O primeiro (figura 6), consiste em uma análise feita a partir do requisito não funcional comunidade do jogo, decompondo esse processo em 2 processos, onde para se obter a funcionalidade de comunidade do jogo é necessário que os Jogadores estejam ativos no jogo. O sistema, no caso o jogo, deve manter a comunicação ativa, para que os jogadores interajam entre si. ou seja, continuem jogando o *game* para manter a comunidade ativa.

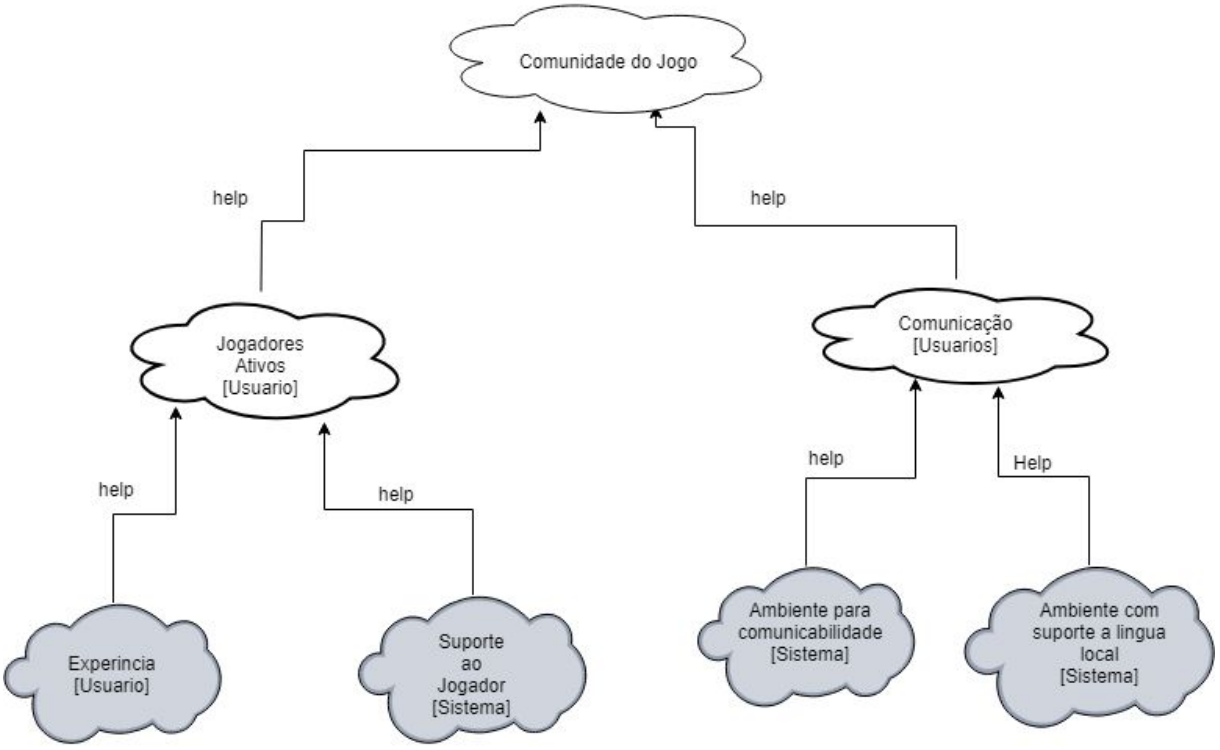

Figura 6: Modelagem *NFR* do processo Comunidade do Jogo

Em jogadores ativos, existe 2 operacionalizações, a experiência do jogador, que é como o jogador se relaciona com o jogo, se passou muito tempo jogando, se conseguiu atingir as expectativas, todas essas sensações que o jogo eletrônico trás para o jogador, influência na operação do usuário em manter ele ativo. A outra operação desse processo, é o suporte que o jogador recebe do sistema, caso ela tenha algum problema e acaba se frustrando com o jogo, é necessário que a empresa ajude o jogador para resolver esses possíveis de desenvolvimento, que afetam a qualidade do produto final.

Um processo muito importante que influencia no requisito não funcional comunidade do jogo, é a Comunicação, pois com a comunidade ativa onde possui um ambiente de comunicação e suporte, a vida útil do jogo fica mais duradoura, pois os usuários estão trocando suas experiências jogadas. A operacionalização Ambiente para comunicabilidade, é onde os jogadores conseguem compartilhar informações, seja por texto, áudio e até videochamada, tudo no ambiente do jogo.

Outra operacionalização da comunidade do jogo, vinda da comunicação é o Ambiente com suporte a língua local. Neste processo, o foco é como os usuários são redirecionados aos ambientes de comunicação da sua região, compartilhando informações sobre suas experiências jogando o jogo eletrônico com temática *RPG*.

O requisito Cenário (Figura 7), é baseado nos mapas de exploração dos jogos eletrônicos de *RPG*. Para se obter um cenário de qualidade, é necessário destacar como a Ambientação e a imersão do jogador influência nesse processo. Ambientação seria a representação do sistema, toda a modelagem do lugar, as texturas, as cores, luminosidade, a fauna e a flora, expressa em computação gráfica.

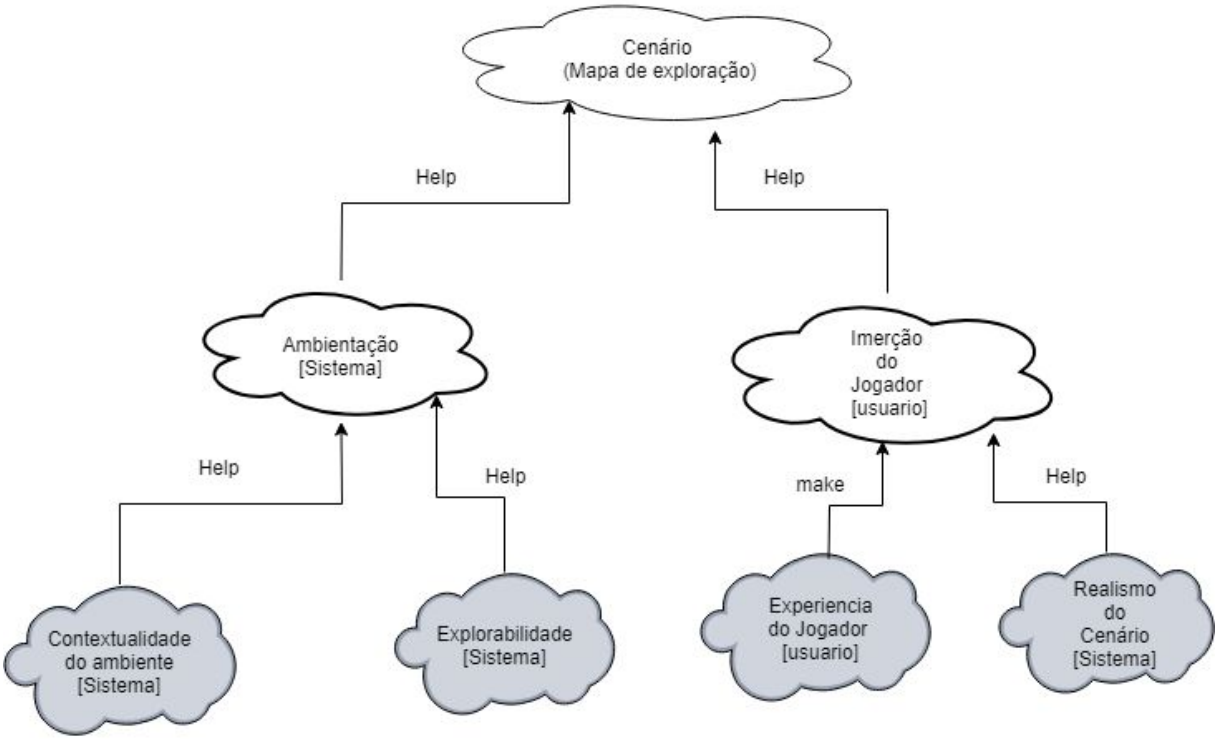

Figura 7: *NFR* Cenário voltado para exploração do mapa.

A decomposição Ambientação necessita de mais duas operacionalizações do próprio sistema para atender a qualidade do requisito. O primeiro é a contextualidade do ambiente, por exemplo, um jogo com um cenário se passa em uma cidade da idade média em algum lugar da europa, o jogo deve apresentar as características do local, todo o contexto da época deve estar presente no *design* do mapa.

O segundo, se trata da Explorabilidade do cenário, como o sistema permite que jogador explora, se o jogador pode interagir com os personagens não jogáveis do mapa, se esses *NPC (Non-Player Character)* vão influenciar na exploração. A explorabilidade deve ser ampla e permitindo que jogador desfrute de todo o cenário, compondo na riqueza do cenário.

Outro processo decomposto do Cenário, é a imersão do jogador, como ele se sente explorando o local, se todo o ambiente faz que o jogador consiga ficar imerso no que o cenário propõe pro jogador, esse requisito é importante, pois convence o jogador que ele está,

por exemplo, explorando uma cidade da idade média. Para que esse requisito seja atendido, é fundamental que o jogador tenha uma certa experiência com os jogos do gênero, representado como *make* na modelagem. Pois se o jogador está acostumado com os cenários de *RPG* a possibilidade dele ter um olhar mais crítico é maior, assim influencia na qualidade a ser atendida.

Para auxiliar na imersão, a operacionalização realismo do cenário ajuda a convencer o jogador que está explorando e vivendo o tema do local, essa operação não está ligada a modelagem 3D do cenário, e sim como ela convence o jogador que o ambiente que está jogando é semelhante ao do mundo real, de acordo com a proposta do jogo.

Por fim, a última modelagem, consiste na velocidade de jogo (Figura 8), nesse caso, nas mecânicas, que são como as funcionalidades dos *softwares*, essas mecânicas é a jogabilidade dos jogos de *RPG* (VIEIRA, 2016). Um Exemplo, existem jogos de *RPG* que possuem mecânicas de batalhas baseados em turnos, o jogador pode atacar, usar itens, defender e fugir da batalha se disponível. Essas opções de combate são as mecânicas do *game*.

Um dos processos destacados, é a legibilidade do sistema, para que o usuário entenda as mecânicas e contribua para velocidade do jogo, é essencial que o sistema seja transparente, para isso acontecer, modelado como *help*, a operação coerência da interface deve ser implementada,Ela consiste em um mapa de ações onde o jogador esteja habituado, devem apresentar lógicas de botões e navegabilidade da interface de forma que possa realizar as interações com o jogo, sem apresentar dificuldade para o jogador.

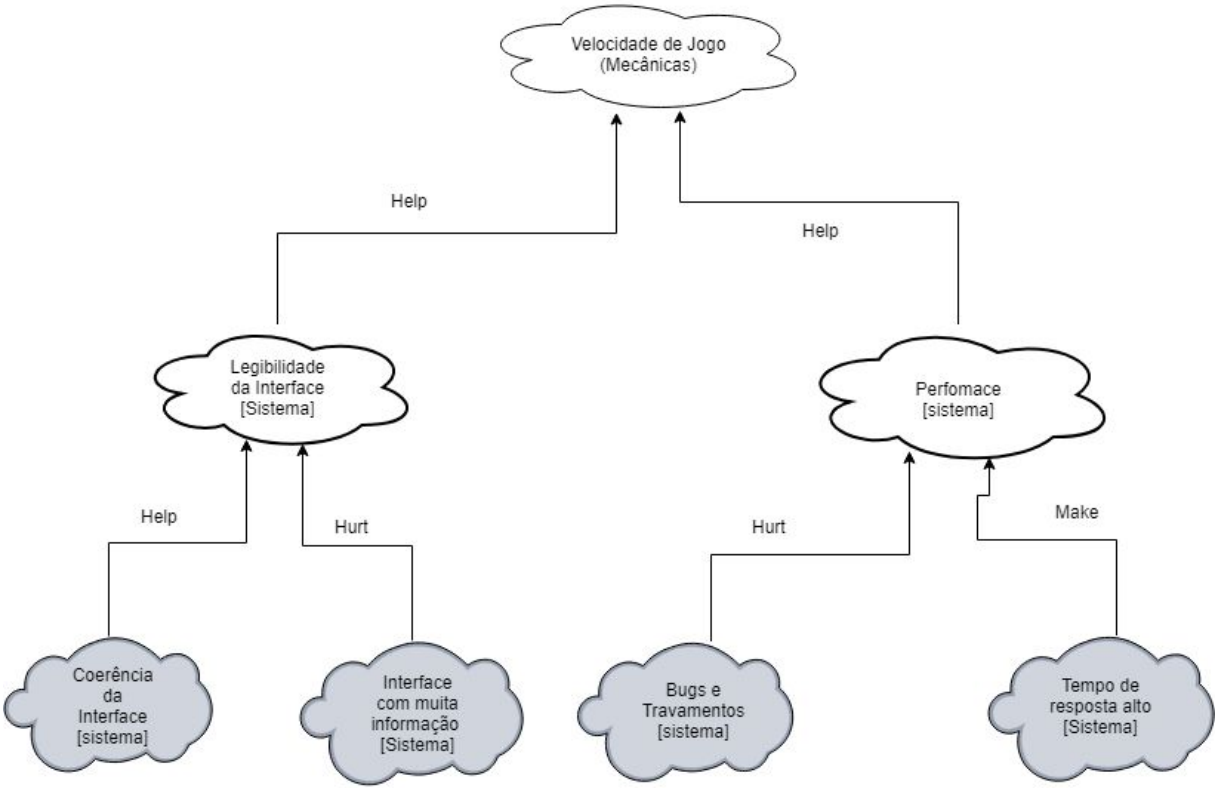

Figura 8: Modelo NFR do requisito não funcional Velocidade de Jogo

A operação Interface com muita informação, representa uma tela onde possui tanta informação que chega desnecessário para jogador, isso pode comprometer a velocidade que o jogador executa as mecânicas, por ter uma tela poluída, a maioria dos jogadores tende a demorar compreender a interface.

O requisito não-funcional, performance provinda do sistema, condiz de como flui as mecânicas o jogo, esse requisito está ligado a otimização do motor gráfico que é usado no jogo em relação ao aparelho eletrônico que vai ser utilizado para rodar o jogo. Se performace atender o necessário para manter a velocidade do jogo, ele irá atender um tempo de resposta alto, uma operacionalização que está essa ligação é representada pela seta *make*.

A operação que é prejudicial à performance, são os Bugs e travamentos do motor gráfico, representado com a ligação *hurt*, se o jogo não representa o desejado para o jogador, ou não executa aquilo que foi determinado no seu requisito, a performance do jogo é comprometida.

6. CONCLUSÃO

Concluímos que o estudo deste trabalho, mostrou a eficiência e importância do modelo *Non-Functional Framework* para a Engenharia de *Software,* no âmbito de jogos de entretenimento com temática *RPG.* A modelagem estabelece as operacionalizações dos requisitos não funcionais e como determina a qualidade do produto, estabelecendo o modelos para uma leitura mais ampla e informatizada.

7. REFERÊNCIAS

PRESSMAN, R. S. Engenharia de software. São Paulo: Makron Books, 2016.

van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: Proceedings of the 5th IEEE international Symposium on Requirements Engineering, August 27-31, 2001, p. 249. IEEE Computer Society, Washington (2001)

Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. International Series in Software Engineering, vol. 5, p. 476. Springer, Heidelberg (1999)

Gartner Inc. - Technology Research http://www.gartner.com/newsroom/id/2614915

UOL. Disponível em: https://jogos.uol.com.br/ultimas-noticias/2018/04/09/gta-v-e-o-produto-mais-lucrativo-da-historia-do-entretenimento.htm

Leite,L.C.;Introdução à História dos Jogos Eletrônicos.Dissertação PUC-RIO.2003

Royce, W.W. Managing the development of large software systems: concepts and techniques. Proc. IEEE Westcon, Los Angeles, CA.

Gilb, T. Principles of Software Engineering Management. Addison-Wesley, 1988.

Brooks, F. No Silver Bullet: Essence and Accidents of Software Engineering. Proc. IFIP, IEEE CS Press, pp. 1069-1076; reprinted in IEEE Computer, pp. 10-19, Apr. 1987.

Vieira, A. L. R. Disponivel em: https://www.fabricadejogos.net/posts/mecanica-de-jogos-parte-1/

YU, E. Modelling Strategic Relationships for Process Reengineering. (Tese de Doutorado) University of Toronto. Toronto, ON, Canada. 1995.