UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Cristiano Politowski

SISTEMA DE RECOMENDAÇÃO DE PROCESSOS PARA O DESENVOLVIMENTO DE JOGOS DIGITAIS

Santa Maria, RS

2017

Cristiano Politowski

SISTEMA DE RECOMENDAÇÃO DE PROCESSOS PARA O DESENVOLVIMENTO DE JOGOS DIGITAIS

Dissertação apresentada ao Curso de Pós-Graduação em Informática, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

Orientadora: Profª. Drª. Lisandra Manzoni Fontoura

Santa Maria, RS

2017

Cristiano Politowski

SISTEMA DE RECOMENDAÇÃO DE PROCESSOS PARA O DESENVOLVIMENTO DE JOGOS DIGITAIS

Dissertação apresentada ao Curso de Pós-Graduação em Informática, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**.

Aprovado em 06 de Abril de 2017:

Lisandra Manzoni Fontoura (UFSM)
(Presidente/Orientadora)

Eduardo Kessler Piveta (UFSM)

Sabrina Marczak (PUCRS)

Santa Maria, RS
2017

À minha mãe.

AGRADECIMENTOS

À minha mãe, porque ela é incrível.

À minha orientadora, professora Dr^{a}. Lisandra Manzoni Fontoura, por ter dado a confiança necessária para realizar esta pesquisa. Obrigado por ser essa pessoa querida por todos.

Ao professor Dr. Fábio Petrillo, cujo trabalho serviu de inspiração para esta pesquisa, além de estar sempre à disposição para esclarecimentos, broncas e conselhos. Definitivamente, este trabalho não existiria sem o seu auxílio.

Aos colegas e amigos que fiz no grupo de pesquisa, na UFSM e em Santa Maria. São pessoas que levarei para sempre como boas lembranças.

À CAPES/CNPQ por ter apoiado financeiramente esta pesquisa.

“Don’t be afraid,
be ready!
(Edward Snowden)

RESUMO

O mercado de jogos digitais é uma indústria bilionária e que possui problemas na forma pela qual os jogos são desenvolvidos. Uma maneira de combater estes problemas é utilizando ferramentas de auxílio aos desenvolvedores, como Sistemas de Recomendação. Estas aplicações possuem um ramo destinado à engenharia de software, auxiliando o desenvolvedor gerando tarefas recomendadas. Este trabalho tem como objetivo mitigar os problemas em projetos de jogos digitais, desenvolvendo um sistema de recomendação de processos, utilizando como base o aprendizado por meio de experiências passadas em projetos de jogos similares, encontrados na forma de *postmortems*. O resultado deste trabalho apresenta três principais contribuições. A primeira delas é uma base de dados contento experiências extraídas de *postmortems* na forma de processos de desenvolvimento. A segunda é a definição de um contexto para projetos de jogos digitais que caracteriza e categoriza os projetos de jogos. Por fim, um sistema de recomendação para projetos de jogos digitais, que utiliza uma lista de projetos similares e o perfil do time de desenvolvimento para gerar um novo processo, com dicas e orientações sobre tarefas a serem executadas. Os processos recomendados auxiliam os desenvolvedores na concepção do processo de desenvolvimento do jogo, listando atividades e práticas de projetos com contexto similar, na forma de um processo de software. Os processos extraídos foram validados diretamente com os desenvolvedores dos jogos. Os elementos dos processos gerados pelo sistema de recomendação foram validados utilizando *precision* e *recall*, e também juntamente com os desenvolvedores/autores dos projetos de jogos analisados.

Palavras-chave: processo de desenvolvimento de software. desenvolvimento de jogos. sistema de recomendação.

ABSTRACT

The digital gaming market is a billion dollar industry that faces issues in the way games are developed. One method to easy these issues is to use developer aid tools such as Recommendation Systems. These applications have a branch for software engineering, assisting the developer by generating recommended tasks. This work, therefore, aims to mitigate issues in digital game projects, developing a processes recommendation system, based on learning through past experiences in similar game projects, found in the form of postmortems. Our work brings three main contributions. The first one is a database containing experiences extracted from postmortems in the shape of development processes. The second is the definition of a context for digital gaming projects that characterizes and categorizes gaming projects. The third is a recommendation system for digital game projects that uses a list of similar projects and the development team profile to generate a new process with tips and guidance on tasks to be performed. Recommended processes assist developers in designing the game development process, listing activities and practices of projects with similar context, in the form of a software process. The extracted processes were validated with the game developers. The elements of the processes, generated by the recommendation system, were validated using precision and recall, and also together with the developers / authors of the analyzed game projects.

Keywords: software development process. game development. recommendation system.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A indústria de jogos digitais é um mercado bilionário cuja receita tem aumentado ao longo dos anos. De acordo com a empresa de marketing digital Newzoo (NEWZOO, 2016), em 2016 esta indústria movimentou cerca de U$99,6 bilhões, 8,5% a mais que o ano de 2015, com uma expectativa de U$118 bilhões em 2019.

O desenvolvimento de jogos é uma atividade de extrema complexidade (GERSHENFELD; LOPARCO; BARAJAS, 2003). Estudos sobre problemas na indústria de jogos concluíram que os projetos de jogos não sofrem de problemas tecnológicos, mas essencialmente de gerenciamento e processo (PETRILLO *et al.*, 2008; PETRILLO *et al.*, 2009). Para amenizar estes problemas, alguns trabalhos vêm abordando o assunto, especialmente sobre a adoção de práticas ágeis no desenvolvimento de jogos (KANODE; HADDAD, 2009; PETRILLO; PIMENTA, 2010; KEITH, 2010). Entretanto, estes trabalhos não abordam o uso sistemático de processos de desenvolvimento no contexto do jogo digital, mas sim apontam as inúmeras falhas na execução do projeto de um jogo (mais detalhes sobre estes trabalhos na Seção 3).

Embora o jogo digital seja software, o mesmo possui características e problemas particulares que aumentam sua complexidade quando comparado ao desenvolvimento de software tradicional (BLOW, 2004; MURPHY-HILL; ZIMMERMANN; NAGAPPAN, 2014). Assim sendo, devido à alta dificuldade no desenvolvimento de jogos, combinada com a multidisciplinaridade dos profissionais envolvidos, alguns autores recomendam o uso de técnicas de Engenharia de Software para gerenciar e desenvolver projetos de jogos (CALLELE; NEUFELD; SCHNEIDER, 2005).

1.1 MOTIVAÇÃO

Conforme mostram os resultados das pesquisas citadas anteriormente, apesar de décadas de existência, o processo de desenvolvimento na indústria de jogos, de forma geral, não evoluiu tanto quanto o desenvolvimento de software tradicional. Através de análises de *postmortems*, Petrillo *et al.* (2009) dignosticaram vários problemas enfrentados pelos desenvolvedores durante o projeto do jogo, como “escopo não realista”, *“feature creep”* e “cortes de *features*”. Somado a isso, Murphy-Hill, Zimmermann e Nagappan (2014) mostraram que a indústria de jogos, por não fazer uso de processos sistemáticos, possui uma falta de maturidade. Fato este evidenciado pelo seguinte relato da experiência de um desenvolvedor, retirado de uma de suas entrevistas (descrito abaixo). Nesta frase, o desenvolvedor evidencia a falta de um processo de desenvolvimento ao confundir métodos ágeis com *Hacking*:

> *"[Desenvolvedores de jogos] operam em um, nós chamamos de, modo Ágil, ou se você preferir, modo hacking... É quase tudo hackeando coisas, mas há muitas pessoas realmente inteligentes aqui, então isso funciona."* Tradução livre de Murphy-Hill, Zimmermann e Nagappan (2014).

Outro fator que evidencia os problemas enfrentados na indústria de jogos é apresentado no relatório anual da IGDA (WESTSTAR; ANDREI-GEDJA, 2016), no qual 52% dos entrevistados responderam "sim" quando perguntados se *crunch time*[1] era uma prática necessária durante o desenvolvimento de um jogo.

Em trabalho prévio (POLITOWSKI *et al.*, 2016a), empresas de desenvolvimento de jogos foram questionadas sobre a forma empregada pela equipe para a construção de seus jogos. Os resultados demonstraram uma falta de sistematização no processo de desenvolvimento bem como pouca maturidade dos desenvolvedores em relação a práticas de engenharia de software.

Para auxiliar os desenvolvedores na adoção e sistematização do processo de desenvolvimento de software, é necessário pensar em suas atividades relacionadas. Uma maneira de melhorar o processo é aprendendo com erros e acertos vividos pelos desenvolvedores em projetos anteriores. Neste caso, a principal fonte de informações são os *postmortems*, artigos escritos após o término do projeto, resumindo as experiências do time de desenvolvimento (mais detalhes sobre *postmortems* na seção 2.4).

No entanto, as informações não estruturadas dos *postmortems* necessitam de um pré-processamento para servirem como fonte de informação. Ainda assim, é preciso, à priori, elucidar o desenvolvedor sobre os possíveis problemas que poderão acontecer durante a fase de desenvolvimento, algo que, como descrito por Petrillo *et al.* (2008), muitas vezes não faz parte do conhecimento (ou interesse) do desenvolvedor. Portanto, é necessário auxiliar o desenvolvedor recomendando um processo, contendo informações sobre atividades realizadas em jogos de contexto parecido. Assim sendo, criar um sistema de recomendação de processos.

Sistemas de recomendação para engenharia de software, diferentes dos sistemas de recomendação tradicionais, auxiliam o desenvolvedor em atividades técnicas provendo informações relacionadas a atividades e não ao perfil do usuário (ROBILLARD *et al.*, 2014)

Um forma de auxiliar os desenvolvedores em suas atividades é fornecer um primeiro passo para a elaboração de um processo a ser executado. Ou seja, recomendar um conjunto de práticas e ações, na forma de um processo, considerando projetos

[1] Na indústria de jogo, o termo *crunch time* refere-se a períodos de sobrecarga de trabalho, normalmente nas semanas antes do fim de prazos (PETRILLO *et al.*, 2008).

de contexto similar. Por fim, realimentar o sistema com cada resultado (processo), evoluindo continuamente na completude e relevância dos processos gerados.

1.2 OBJETIVO GERAL E OBJETIVOS ESPECÍFICOS

O objetivo geral deste trabalho é desenvolver um sistema de recomendação de processos de software para jogos digitais. O sistema irá utilizar o grau de similaridade do contexto de projetos de jogos, extraídos a partir da análise dos relatos de experiências passadas, na forma de *postmortems*.

Este sistema de recomendação parte de dois princípios primários: **aprender com projetos semelhantes** através da análise de relatos de experiências (*postmortems*) em projetos de jogos, escritos pelos próprios desenvolvedores, e **customizar o processo** de acordo com as características do time, possibilitando que o time defina a forma com que o processo é gerado.

Os objetivos específicos são os seguintes:

- Criar uma base de dados de processos de software extraídos de projetos de jogos digitais (apresentado no Capítulo 5).
- Criar uma forma automatizada de extrair o processo utilizado no projeto a partir da análise de *postmortems* (apresentado no Capítulo 5).
- Elencar elementos de contexto específicos para o desenvolvimento de jogos. Traduzindo em algumas características o processo de desenvolvimento (apresentado na Seção 6.1).
- Criar um sistema de recomendação que leve em consideração as características dos projetos similares e a forma como o time gostaria de desenvolver o jogo (apresentado Capítulo 6).
- Validar os processos gerados pelo sistema de recomendação juntamente com os desenvolvedores dos jogos e autores dos *postmortems* (apresentado no Capítulo 7).

1.3 ESTRUTURA DO TRABALHO

O Capítulo 2 apresenta os conceitos essenciais para o entendimento da pesquisa. O Capítulo 3 descreve os trabalhos relacionados e os artigos publicados durante a pesquisa. O Capítulo 4 mostra o método de desenvolvimento utilizado durante o trabalho. Os Capítulos 5 e 6 apresentam o desenvolvimento do trabalho juntamente com uma seção de discussão. O Capítulo 7 demonstra as validações dos resultados. Por fim, o Capítulo 8 com as considerações finais e resumo das contribuições.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos necessários para o entendimento do restante do texto. A Seção 2.1 apresenta os conceitos de sistema de recomendação tradicional e voltados à engenharia de software. A Seção 2.2 trata dos conceitos de processo de software uma vez que este é o tema principal do trabalho. A Seção 2.4 explica o que é um *postmortem* e também porque ele é uma fonte valiosa de informação sobre o desenvolvimento de jogos e software tradicional. Esta seção é fundamental para o correto entendimento do Capítulo 5. A Seção 2.3 descreve o conceito de contexto de software segundo alguns autores área. Esta seção é essencial para o entendimento do contexto de jogos criado no Capítulo 6.1. A Seção 2.5 apresenta o básico para o entendimento do método *PCA* e do gráfico *biplot*, ambos necessários para o correto entendimento do algoritmo de similaridades apresentado na Seção 6.2.

2.1 SISTEMAS DE RECOMENDAÇÃO

O desenvolvimento de software vem se tornando uma atividade que usa e produz uma vasta quantidade de conhecimento (*knowledge-intensive*). Dominar uma linguagem de programação não é mais suficiente para desenvolver software de forma proficiente, uma vez que desenvolvedores são expostos constantemente a novas tecnologias, componentes e ideias, sem falar de sistemas em constante expansão com adição de novas bibliotecas e recursos. Esta grande quantidade de dados excede a capacidade de assimilação do desenvolvedor, sendo necessário uma forma de navegar por esta informação (ROBILLARD *et al.*, 2014).

Sistemas de recomendação são aplicações com o objetivo de auxiliar usuários em suas decisões, usando uma grande quantidade de informação. Estas aplicações recomendam itens do interesse dos usuários baseando-se em suas preferências, expressa de forma explícita ou implícita. O crescente aumento do volume e complexidade das informações tornou estes sistemas essenciais para usuários. Sistemas de recomendação ajudam a superar o problema de sobrecarga de informação expondo aos usuários os itens mais interessantes e oferecendo surpresa, relevância e diversidade (ROBILLARD; WALKER; ZIMMERMANN, 2010).

Assim como sistemas de recomendação de *websites* ajudam expondo aos usuários itens antes desconhecidos, sistemas de recomendação podem ser usados em domínios técnicos trazendo à tona informações desconhecidas previamente, auxiliando desenvolvedores em suas tarefas. Estes sistemas, denominados sistemas de recomendação para engenharia de software (*Recommendation Systems for Software*

Engineering (RSSEs)), são aplicações que oferecem informações para uma atividade de engenharia de software, na esperança de terem valia para os profissionais envolvidos (ROBILLARD *et al.*, 2014).

Segundo Robillard *et al.* (2014), de forma geral, a arquitetura para a construção de um sistema de recomendação possui quatro fases: pre-processamento dos dados, captura do contexto, produção das recomendações e apresentação das recomendações.

- **Pré-processamento dos dados:** o domínio da engenharia de software demanda um esforço necessário para transformar dados não tratados (*raw data*) em um formato que possa ser interpretado. Aliado a isto, outras atividades de pre-processamento de dados podem acontecer como detecção de dados discrepantes e troca de amostras com valores nulos.

- **Captura de contexto:** enquanto que em domínios tradicionais, como em *e-commerce*, sistemas de recomendação são fortemente dependentes de perfis de usuários (*user-centric*); na engenharia de software, o conceito central é a tarefa (*task-centric*). Há, no entanto, um paradoxo quanto ao grau de precisão da informação, fonte do sistema de recomendação — quanto mais precisa for a informação, mais precisa a recomendação pode ser, mas menos provável que o desenvolvedor precise desta recomendação, visto que a informação já está ali de forma completa e estruturada. Por outro lado, um desenvolvedor com necessidade de um auxílio pode não ter informações suficientes para que o sistema recomende algo útil. Desta forma, sistemas de recomendação geralmente terão como resultado tarefas incompletas ou ruidosas.

- **Produção das recomendações:** uma vez os dados pre-processados e uma quantia suficiente de tarefas de contexto estejam prontos, os algoritmos podem ser executados. Estes são ligados ao domínio do problema e/ou a criatividade do *designer*. No entanto, algoritmos de recomendação comumente utilizados em sistemas tradicionais não costumam funcionar na engenharia de software.

- **Apresentação das recomendações:** normalmente a apresentação das recomendações são listas de itens potencialmente interessantes ao desenvolvedor como funções, classes, relatórios, entre outros.

2.2 PROCESSO DE SOFTWARE

Humphrey (1988) detalha os conceitos que norteiam o tema processo de software, da seguinte forma:

- **Engenharia de software**: A disciplina aplicada da engenharia, científica e de princípios matemáticos, métodos e ferramentas para a produção econômica de um software de qualidade.
- **Processo de engenharia de software**: O conjunto total das atividades de engenharia de software necessárias para transformar requisitos do usuário em software.
- **Arquitetura de processo de software**: Um *framework* no qual processos de software para projetos específicos são definidos.
- **Modelo de processo de software**: Uma instância específica de uma arquitetura de processo de software.
- **Processo de software**: O conjunto de atividades, métodos e práticas que são usadas na produção e evolução do software.

A área de processos de software, como disciplina autônoma, iniciou-se em 1980, a partir de uma série de eventos e workshops sobre o assunto, dentre eles destaca-se o *International Software Process Workshop*[1]. Foram criadas também instituições nos EUA e na Europa para estudar processos de software, tais como: *Software Engineering Institute (SEI)* e *European Software Institute (ESI)* (FUGGETTA, 2000).

Um dos conceitos mais sucintos é o de Sommerville (2011), que vê o processo de software como um conjunto de atividades e resultados associados que geram um produto de software.

Armbrust *et al.* (2009) dá destaque aos requisitos e ao contexto, para ele, um processo de software é visto como um conjunto completo de atividades de engenharia de software necessárias para transformar os requisitos de um usuário em um produto de software considerando um contexto.

Fuggetta (2000) define processo de software como o conjunto coerente de políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos que são necessários para conceber, desenvolver, implantar e manter um produto de software.

2.2.1 A abordagem preditiva e o advento dos métodos ágeis

Criar software utilizando processos se tornou comum após os crescentes problemas enfrentados pelos desenvolvedores que culminaram com a alegada “crise do software” em 1968 (NAUR; RANDELL, 1969).

Existem diferentes tipos de processos, alguns compartilham conceitos e podem ser usados simultaneamente, como *Extreme Programming* (BECK, 2000) e *Scrum*

[1] Houveram apenas três eventos: 88, 90 e 96 — http://dl.acm.org/event.cfm?id=RE359

(SCHWABER; BEEDLE, 2002); enquanto outros diferem em suas bases, como *Waterfall* (ROYCE, 1970) e *Lean* (POPPENDIECK; POPPENDIECK, 2003). Não há, porém, uma separação clara dos tipos de processos. Neste trabalho os processos serão separados em duas categorias, nomeados de **preditivo** e **adaptativo** (PRESSMAN, 2005).

Processos preditivos possuem um fluxo sequencial de atividades onde a próxima fase só começa após o término da anterior, culminando em uma entrega de valor todo de uma vez. Para tanto, requer uma forte avaliação dos requisitos e procedimentos de resolução de problemas (ROYCE, 1970). *Waterfall* é um exemplo deste tipo de processo.

Por outro lado, processos adaptativos possuem ciclos curtos de desenvolvimento realizados repetidamente, entregando um componente (*feature*) pronto para ser usado a cada término (LARMAN; BASILI, 2003). O foco principal destes métodos de desenvolvimento é melhorar continuamente o processo e suas práticas (FOWLER, 2001). Exemplos destes tipos de processos são *Scrum* e *Extreme Programming*.

Alguns autores, como Larman e Basili (2003), atestam que práticas ágeis são utilizadas desde os primórdios do desenvolvimento de software. Outros, como Fowler (2001), enxergam uma evolução natural no modo em que se desenvolve software. Para o autor, processos planejados são oriundos de áreas como engenharia civil, na qual o profissional define, por exemplo, todo o projeto do prédio e suas implicações, antes de de sua execução, visto que é mais custoso e perigoso fazer alterações durante a construção. Estas práticas foram empregadas na construção de software, resultando em processos de desenvolvimento muitas vezes burocráticos e pouco suscetíveis a mudanças. Ainda segundo o autor, este cenário é justamente o oposto ao encontrado no desenvolvimento de aplicações, na qual a mudança é uma certeza.

Esta forma de encarar a maneira de desenvolver software tem ganho rápida adoção por parte da comunidade de desenvolvedores desde a criação do manifesto ágil[2] em meados de 2001. Segundo o relatório anual, realizado pela *Version One* (VERSIONONE, 2015), em 2015, apenas 4% dos respondentes afirmaram não usarem práticas ágeis em algum time. Este fato também é evidenciado na área de jogos. Segundo Petrillo e Pimenta (2010), desenvolvedores de jogos utilizam práticas ágeis mesmo que de forma involuntária. Ademais, pesquisas com *postmortems* mostram que as metodologias ágeis já são maioria no desenvolvimento de jogos (POLITOWSKI *et al.*, 2016a; MUSIL *et al.*, 2010).

[2] O Manifesto Ágil (http://agilemanifesto.org) foi resultado da reunião de profissionais cansados do modo tradicional de desenvolvimento de software. Neste documento eles definem alguns princípios para o desenvolvimento, embora não definam qualquer tipo de processo.

2.3 CONTEXTO DE SOFTWARE

O contexto em que o software vai ser desenvolvido, ou seja, seus requisitos, características do time de desenvolvimento, orçamento, prazo, entre outros, modificam a forma de desenvolver as atividades durante o processo de desenvolvimento. Fuggetta e Nitto (2014) dão ênfase ao contexto e na necessidade de adaptação e extensão dos padrões de qualidade, em suas palavras:

> *"De forma geral, a variedade de situações e contextos na qual o software é usado está expandindo significativamente. Consequentemente, modelos e padrões de qualidade existentes precisam ser estendidos e adaptados para diferentes situações e contextos." Tradução livre de Fuggetta e Nitto (2014)*

Kruchten (2013), Boehm e Turner (2003) citam o contexto do projeto como fator principal na escolha de processos de software. Seguindo esta linha de pensamento, entende-se que nenhum processo é melhor que outro, pois há vários fatores que envolvem o desenvolvimento de software e, para jogos, esta gama de características é ainda mais variada, dada a maior multidisciplinaridade dos times e atividades envolvidas.

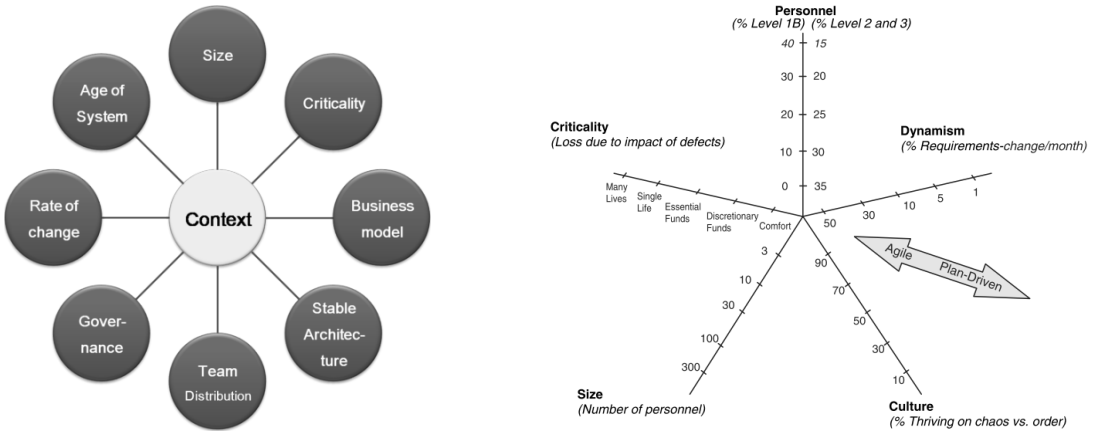
Kruchten (2013) dá forte ênfase à importância do contexto no processo de desenvolvimento de software (Figura 1a). O autor definiu um modelo, denominado *Octopus*, para contextualizar projetos de software, definindo características para projetos ágeis (*agile sweet spot*). O modelo *Octopus* traz 8 fatores contextuais em nível de projeto:

1. **Tamanho:** tamanho do sistema;
2. **Arquitetura estável:** arquitetura base para iniciar a construção do sistema;
3. **Modelo de negócio:** relacionado aos recursos financeiros;
4. **Distribuição do time:** número e localização dos times envolvidos;
5. **Taxa de mudança:** estabilidade do ambiente de negócio;
6. **Idade do sistema:** novo sistema ou evolução/extensão de legados;
7. **Criticalidade:** o quanto uma falha no sistema pode afetar alguém;
8. **Governança:** questões de gerenciamento.

Em seu livro, Boehm e Turner (2003) definem cinco fatores de contexto em projetos de software que distinguem processos planejados (*plan-driven methods*) de

processos ágeis, são eles: “tamanho do time” (*size*), “criticalidade do sistema” (*criticality*), “habilidade dos profissionais” (*personnel*), “dinamismo” ou taxa de mudanças (*dynamism*) e “cultura do time” (*culture*). Conforme a Figura 1b, quanto mais ao centro do gráfico os valores, mais adequado a métodos ágeis é o projeto. Do contrário, quanto mais nas extremidades os valores estejam, mais voltados à projetos planejados.

Figura 1 – Duas abordagens para definir o contexto de projetos de software.



(a) Contexto por Kruchten (2013). (b) Contexto por Boehm e Turner (2003).

Fonte: (KRUCHTEN, 2013) e (BOEHM; TURNER, 2003).

No âmbito do software tradicional e, principalmente, com metodologias ágeis, estes exemplos “encaixam” ou, no pior dos casos, servem de base para uma extensão ou nova definição. Quando o tema é a área de jogos, as variáveis definidas não abrangem a quantidade de caraterísticas de forma suficiente. Existem diversos outros detalhes, como descrito na Seção 6.1, em que características podem definir a forma na qual o jogo vai ser desenvolvido.

2.4 *POSTMORTEM* DE JOGOS

O termo *postmortem* designa um documento que resume as experiências no desenvolvimento do projeto, com forte ênfase nos aspectos positivos e negativos (HAMANN, 2003). Normalmente é feito logo após o término do projeto, por gerentes ou desenvolvedores sênior (CALLELE; NEUFELD; SCHNEIDER, 2005). É uma ferramenta importante para gestão do conhecimento (FRIDLEY, 2013), possibilitando ao time aprender com suas próprias experiências e planejar projetos futuros. A análise de *postmortems* pode ser tão reveladora que alguns autores (FRIDLEY, 2013) argumentam que nenhum projeto deveria terminar sem um *postmortem*.

Postmortems são muito utilizados na indústria de jogos. Alguns *websites* sobre jogos devotam seções inteiras para estes documentos, como *Gamasutra*[3] e *Gamedev*[4].

[3] http://www.gamasutra.com
[4] http://www.gamedev.net

Estes *postmortems* pertencem a uma variedade de perfis e projetos de desenvolvimento, abrangendo desde poucos desenvolvedores em projetos curtos até dezenas de desenvolvedores em projetos com mais de cinco anos.

Os *postmortems* publicados pelo *Gamasutra* em sua maioria seguem a estrutura proposta pela *Open Letter Template* (MYLLYAHO *et al.*, 2004), a qual é composta por três seções. A primeira resume o projeto e apresenta os resultados mais importantes. As duas seções seguintes discutem as questões mais interessantes do desenvolvimento do jogo:

- ***What went right:*** discute as melhores práticas adotadas pelos desenvolvedores, soluções, melhorias e decisões de gerenciamento que tenham aumentado a eficiência do time. Todos estes aspectos são elementos críticos a serem usados no planejamento de projetos futuros.
- ***What went wrong:*** discute dificuldades, armadilhas e erros vividas pelo time de desenvolvimento no projeto, tanto técnicos quanto gerenciais.

A informação contida nos *postmortems* constitui uma peça de informação que pode ser reusada por qualquer time de desenvolvimento, incluindo exemplos e experiências reais de desenvolvimento. Eles permitem compartilhamento de conhecimento e são úteis para planejar futuros projetos ou melhorar processos. Um exemplo de *postmortem*, do jogo *"Slow Down, Bull"*, está disponível no Anexo A.

2.5 *PRINCIPAL COMPONENT ANALYSIS (PCA)*

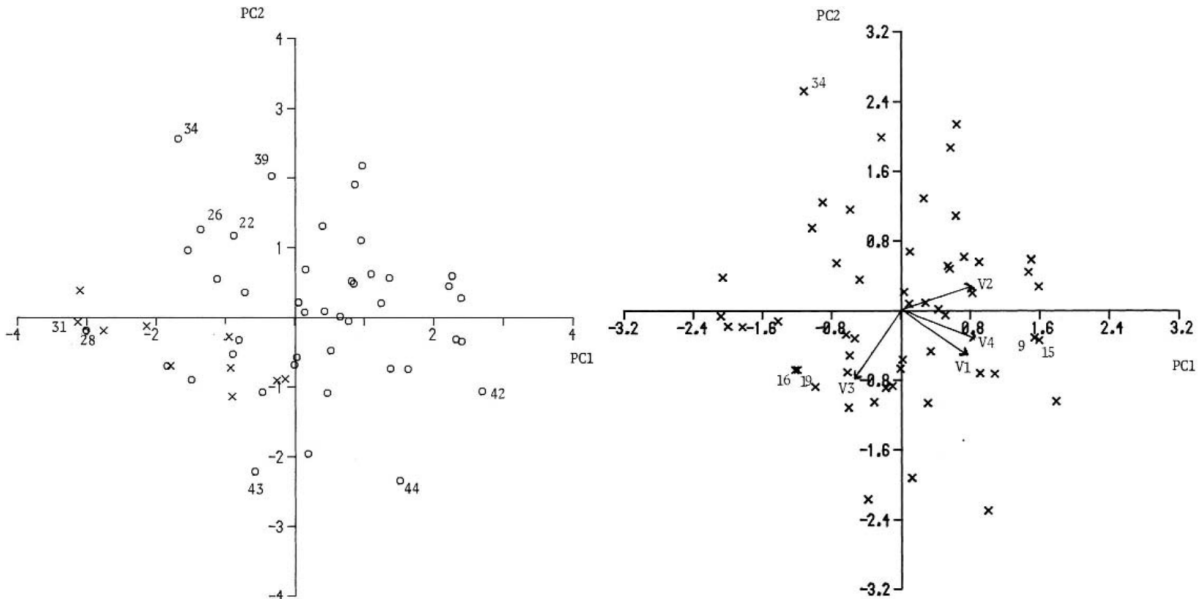
A ideia central do *principal component analysis (PCA)* é reduzir a dimensionalidade de um conjunto de dados consistido de um grande número de variáveis relacionadas, mantendo a maior parte da variação contida no dados. Isto se dá transformando para um novo conjunto de variáveis, os componentes principais (*principal componenets (PCs)*), que são não-relacionados e ordenados da forma que os primeiros tenham a maior variação presente em todas as variáveis originais (JOLLIFFE, 2002).

PCA é especialmente útil quando há um conjunto grande de variáveis. Neste caso os dados podem ser plotados em duas dimensões, possibilitando uma representação visual direta da aparência dos dados, ao invés de números difíceis de interpretar. Uma plotagem dos dois primeiros *PCs* fornece a melhor representação possível dos dados em duas dimensões além de serem particularmente úteis para detectar padrões nos mesmos (JOLLIFFE, 2002).

Para representar graficamente os dados em um plano 2D existem algumas soluções, entre as quais o *biplot*. Segundo Jolliffe (2002), *biplots* além de plotar as N

amostras em duas dimensões, permite plotar também as variáveis definidas, possibilitando o cruzamento de informações. Por exemplo, a Figura 2 ilustra dois exemplos de plotagem PCA. A Figura 2a é uma plotagem dos dois primeiros *PCs* enquanto que a Figura 2b é um *biplot* dos dados, mostrando os vetores que representam as variáveis: V1, V2, V3 e V4.

Figura 2 – Plotagem de PCs em duas dimensões (JOLLIFFE, 2002).



(a) Plotagem normal de dois PCs. (b) *Biplot*.

Fonte: adaptação de Jolliffe (2002).

O mais importante, em ambas as figuras, é que elas representam o grau de similaridades entre as amostras, considerando as quatro variáveis. Ou seja, quanto mais próximo uma amostra está da outra, maior a correlação entre seus valores. Entretanto, vale notar que o ganho do *biplot* está na habilidade de nomear os vetores, dando a noção do valor da amostra junto a variável. Por exemplo, analisando os vetores V1 e V4 constata-se que, como estão próximos, possuem uma correlação positiva alta, enquanto que os vetores V2 e V3, como estão em quadrantes opostos, possuem uma correlação negativa.

2.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou os principais conceitos necessários para um melhor entendimento do trabalho. Primeiramente foi apresentado o conceito de sistemas de recomendação tradicional e de sistemas voltados para engenharia de software. Estes fazem parte de uma sub-área de sistemas de recomendações que focam na geração de informação relevante aos desenvolvedores (*stakeholders*) na forma de tarefas (*tasks*).

Estas informações não são apresentadas de forma completa, pois o contexto de cada aplicação varia significantemente, não permitindo o uso integral de uma solução.

Outro conceito apresentado foi o de “processo de software” e suas terminologias correlatas, segundo alguns autores, definindo o conceito de Humphrey (1988) como o que será utilizado neste trabalho, ou seja, um “processo de software é um conjunto de atividades, métodos e práticas utilizadas na produção e evolução do software”. Ainda sobre o tema, foi realizada uma breve discussão sobre as diferenças entre as abordagens “preditivas” (processos planejados) e “adaptativas” (processos ágeis), definindo as duas categorias de processo utilizadas neste trabalho.

Neste capítulo também foi descrito o conceito de contexto de software segundo alguns autores como Kruchten (2013), Boehm e Turner (2003) e suas abordagens para contextualizar projetos de software tradicional.

Foi apresentado o conceito de *postmortem*, sua estrutura e propósito. *Postmortems* são a principal fonte de informações sobre projetos de jogos. Embora sua estrutura seja livre, é uma ferramenta que pode ser utilizada para revisão e planejamento de ações futuras.

Por fim, o conceito da técnica de *principal component analysis*, ou somente *PCA*, foi explicada. Esta técnica, juntamente com a plotagem *biplot*, permite a visualização da correlação entre amostras, em um plano 2D, em grandes conjuntos de variáveis.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados ao tema da pesquisa que serviram como ponto-de-partida para este projeto. Ressalta-se também a substancial colaboração de alguns autores que enviaram dados, explicações e informações extras de seus projetos.

O primeiro trabalho a se destacar, e o que formou a base para esta pesquisa, é o de Petrillo *et al.* (2009). Neste trabalho os autores analisaram *postmortems* e listaram os problemas mais comuns enfrentados por desenvolvedores de jogos, conforme a Tabela 1. Os autores concluíram que a indústria de desenvolvimento de jogos não sofre de problemas de nível tecnológico, mas sim de gerenciamento. Também destaca que todos os principais problemas da indústria tradicional de software são encontrados na indústria de jogos, ou seja, estão relacionados. Por fim, o autor comparou as duas indústrias, destacando similaridades na frequência de alguns problemas como atrasos e requisitos.

Tabela 1 – Problemas mais comums no desenvolvimento de jogos.

Problema	**Frequência**
Unrealistic scope	75%
Feature Creep	75%
Cutting features	70%
Design problems	65%
Delays	65%
Technological problems	60%
Crunch time	45%
Lack of Documentation	40%
Communication problems	35%
Tool problems	35%
Test problems	35%
Team building	35%
Number of defects	30%
Loss of Professionals	25%
On Over Budget	25%

Fonte: adaptado de Petrillo *et al.* (2009).

Os mesmos autores, em um trabalho complementar (PETRILLO; PIMENTA, 2010), listaram as práticas mais comuns em projetos de jogos. Um dos resultados do trabalho mostrou que, mesmo informalmente, os times de desenvolvimento de jogos estão adotando práticas ágeis e, por esta razão, a implantação de métodos ágeis como *Scrum* e *XP* podem ocorrer de forma natural.

Murphy-Hill, Zimmermann e Nagappan (2014) realizaram 14 entrevistas qualitativas e 364 quantitativas com profissionais da área de desenvolvimento de jogos. Entre

vários resultados, destacaram a falsa adesão de métodos ágeis, ou seja, desenvolvedores que citam *agile* para justificar a falta de processo. Para os autores, talvez a razão dos times de desenvolvimento não usarem processos é a noção de que a imposição de controle vá contra o principal princípio de desenvolvimento: a criatividade.

Os autores também destacaram os cronogramas (*schedules*) como sendo muito mais curtos quando comparados ao desenvolvimento de software tradicional. Bem como prazos (*deadlines*) inflexíveis:

> *"Eu acho que talvez parte disso é que prazos normalmente são muito apertados e há um senso de que não se pode perder nenhuma hora de trabalho. De fato, muitos times colocam seus desenvolvedores em prazos incrivelmente apertados."* Tradução livre de Murphy-Hill, Zimmermann e Nagappan (2014).

Por fim, os resultados do trabalho dos autores (MURPHY-HILL; ZIMMERMANN; NAGAPPAN, 2014) sugerem que jogos possuem diferenças significantes ao desenvolvimento de software tradicional.

Jr *et al.* (2016) também trabalharam com *postmortems* de jogos, analisando 155 destes artigos, destacando características e armadilhas no desenvolvimento (*pitfalls*). Como resultado, definiram um conjunto de boas práticas para desenvolvedores de jogos.

O'Hagan, Coleman e O'Connor (2014) realizaram uma revisão sistemática da literatura com 404 artigos analisados, sendo 73% acadêmicos e 27% da indústria. Os resultados mostraram um total de 356 processos de software, agrupados em 23 modelos, dos quais 47% eram de natureza ágil e 53% híbridos (junção de métodos ágeis com tradicionais).

O'Hagan e O'Connor (2015), no ano seguinte, realizaram um estudo de caso da indústria de jogos a fim de investigar o papel de processo de software no desenvolvimento (O'HAGAN; O'CONNOR, 2015). A pesquisa ressaltou vários *gaps* no que tange processos de desenvolvimento de jogos. Segundo os autores, não há um modelo de melhores práticas para desenvolvimento de jogos. Portanto, um modelo de boas práticas poderia ser benéfico para a indústria de jogos, reduzindo o *time-to-market* e também melhorando a qualidade final do produto. Por fim, o trabalho sugere criar tal modelo baseado em padrões já existentes, como a *ISO/IEC 29110*.

Kanode e Haddad (2009) fizeram um trabalho no qual investigaram o papel da engenharia de software na resolução de problemas no desenvolvimento de jogos. Para os autores, o desenvolvimento de jogos possui características únicas e engenharia de software pode ajudar a superar as dificuldades. Para tanto, métodos ágeis podem

ser aplicados na pré-produção para agilizar o processo. Por fim, destacam também a importância do gerenciamento de grandes quantidades de componentes.

Os trabalhos aqui relacionados mostram os problemas e características da indústria de jogos, mais precisamente, problemas enfrentados pelos times de desenvolvimento durante o projeto. No entanto, após a revisão bibliográfica, não há menção de uma implementação com validação de um sistema que visa auxiliar desenvolvedores, recomendando um conjunto de atividades, na forma de um processo, baseando-se em projetos de contexto similar.

3.1 ESTUDOS PRÉVIOS

Durante a pesquisa, alguns trabalhos foram realizados no intuito de, primeiro, investigar a área de desenvolvimento de jogos e, segundo, obter *feedback* de algumas hipóteses.

O primeiro trabalho, de título ***"Software Engineering Processes in Game Development: a Survey about Brazilian Developers' Experiences"***, publicado na *SBGAMES 2016* (POLITOWSKI *et al.*, 2016b), consistiu-se de uma pesquisa online (*survey*) feita com 58 desenvolvedores de jogos brasileiros, onde foram questionados sobre os problemas enfrentados no desenvolvimento de jogos e a relação com a engenharia de software.

Considerando o contexto brasileiro, os dados apresentaram três resultados principais. O primeiro mostra que projetos que usaram abordagens sistemáticas, independente do tipo, resultaram em produtos melhores. O segundo resultado destaca atrasos, escopo não realista e falta de documentação como os problemas mais frequentes enfrentados por desenvolvedores de jogos. Por fim, o artigo descreve algumas considerações coletadas de desenvolvedores e literatura, servindo como uma fonte de conhecimento bem como caracterização dos desenvolvedores brasileiros.

Outra pesquisa realizada, de título ***"Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry"***, publicado no evento *Games and Software Engineering de 2016* (POLITOWSKI *et al.*, 2016a), consistiu de uma coleta, análise e classificação de processos extraídos de *postmortems*. Oriundos do site *Gamasutra*, 20 artigos foram analisados no intuito de traduzir em um processo gráfico as informações descritas pelos desenvolvedores.

O artigo tem como contribuição a metodologia de análise de *postmortems* utilizada para identificar e extrair os processos dos relatos de projetos. O resultado principal do estudo mostra que as antigas práticas de desenvolvimento estão desaparecendo aos poucos, visto que práticas iterativas estão em ascensão e correspondem a pelo menos 65% dos projetos analisados, dos quais 45% explicitamente adotaram práticas

ágeis. Entretanto, processo preditivo ou *waterfall* ainda é usado, correspondendo a pelo menos 30% dos projetos analisados.

3.2 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foram apresentados os trabalhos dos autores que pesquisam sobre desenvolvimento de jogos e processos de software. Os resultados das pesquisas demonstram os problemas enfrentados pela indústria de jogos digitais como um todo. Alguns ainda propõem soluções, como boas práticas a serem utilizadas em futuros projetos. No entanto, não há qualquer menção de implementação e validação de um sistema para auxiliar o desenvolvedor utilizando a recomendação de um conjunto de atividades e práticas (processo) baseadas em projetos com um contexto semelhante. Portanto, sendo os problemas graves, a necessidade de investir em uma solução que os amenize é iminente.

Ainda, neste capítulo são apresentados, de forma sucinta, os dois artigos publicados durante esta pesquisa. O primeiro artigo serviu como investigação da área de desenvolvimento de jogos enquanto o segundo serviu como protótipo para este trabalho.

4 MÉTODO DE PESQUISA

Este capítulo detalha a sequência de passos, bem como suas justificativas, necessários para atingir o objetivo proposto nesta pesquisa. Conforme ilustra a Figura 3, o método utilizado foi dividido em três etapas: extração dos processo de desenvolvimento utilizando *postmortems*, descrito na Seção 4.1; criação do sistema de recomendação de processos, detalhado na Seção 4.2; e a validação do sistema e resultados, descrito na Seção 4.3.

4.1 EXTRAÇÃO DE PROCESSOS DOS *POSTMORTEMS*

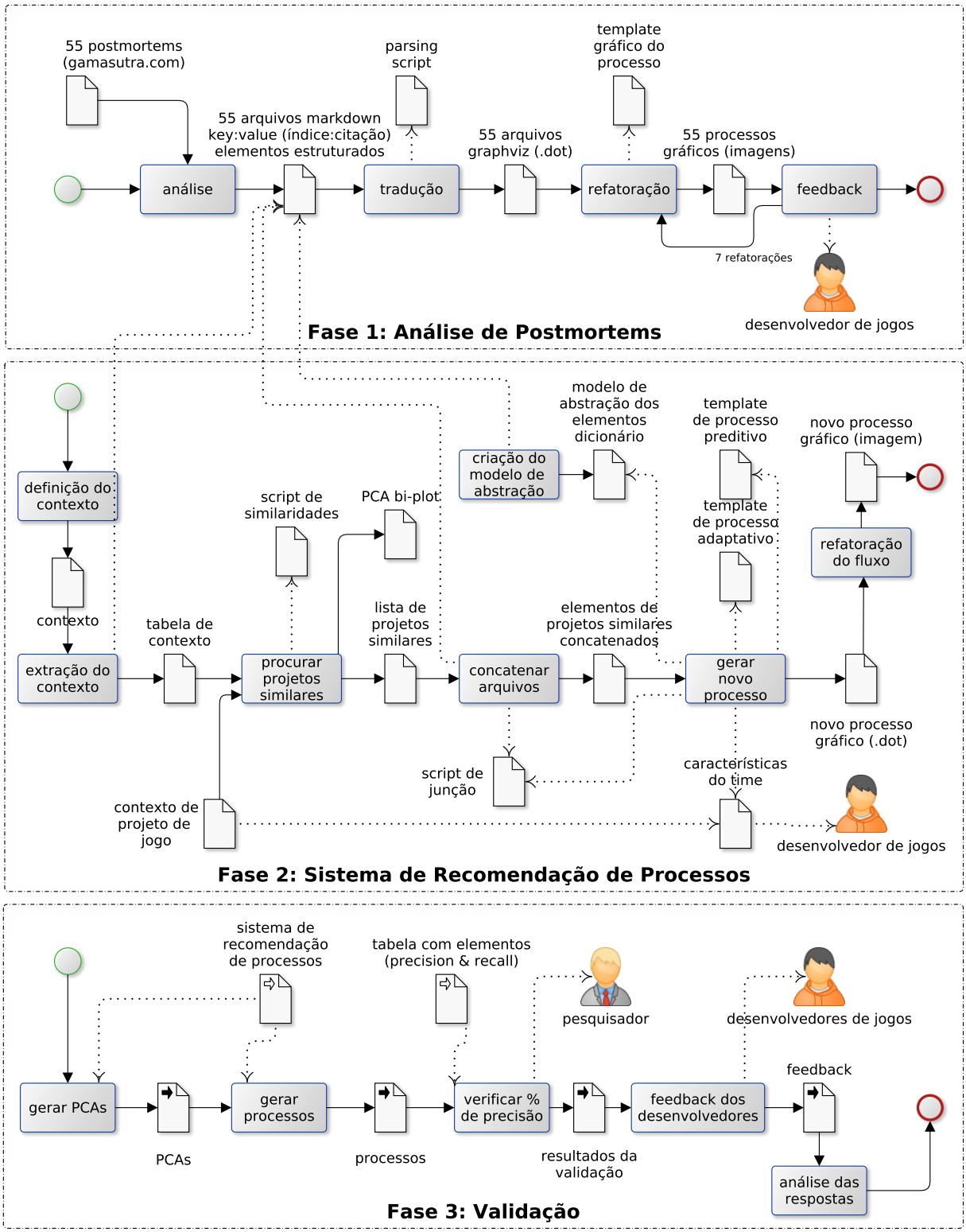
A primeira etapa (fase 1 da Figura 3), apresentada no Capítulo 5, consistiu na coleta e análise dos *postmortems*, extração dos elementos de processos, criação dos processos gráficos e aquisição do *feedback* dos autores (desenvolvedores de jogos).

Inicialmente realizou-se as análises dos relatos dos desenvolvedores de jogos, na forma de *postmortems*, **extraindo as informações que dizem respeito ao processo de desenvolvimento do jogo**, excluindo qualquer outra que não entre neste requisito. Os dados foram extraídos para uma estrutura padrão, a fim de possibilitar a análise automática por parte de um algoritmo. Os *postmortems* possuem muitas informações que não dizem respeito ao processo em que o jogo foi desenvolvido, como detalhes do design e aspectos criativos. Por esta razão, foi necessário separar os dados relevantes que formaram o processo gráfico. Estes dados continham informações relevantes ao processo, como atividades realizadas, detalhes do time de desenvolvimento, práticas utilizadas, ferramentas, prazos e demais informações que passíveis de serem generalizadas para outros projetos.

Com as informações extraídas e estruturadas, um *script* leu cada um dos arquivos separadamente gerando um novo arquivo gráfico, utilizando um modelo gráfico padrão para deixar os processos uniformes. A criação destas representações gráficas do processo de cada *postmortem* analisado demandou muito esforço. No entanto, como esta tarefa foi quase totalmente automatizada, foi possível ter mais precisão e padronização. Ainda, o modelo gráfico padrão permitiu uma análise mais consistente dos processos gerados.

Com os modelos gráficos dos processos extraídos, entrou-se em contato com alguns desenvolvedores dos jogos e autores dos *postmortems* para requisitar um *feedback* sobre o processo final. Com as respostas foram realizadas refatorações nos processos para que os mesmos ficassem em conformidade com o que aconteceu de fato durante o desenvolvimento do jogo. Ainda que de forma semi-automatizada, houve

Figura 3 – Etapas do método de desenvolvimento do projeto.



Fonte: elaborado pelo autor.

o viés de análise nos processos extraídos, ou seja, a interpretação do ser humano. Por este motivo, a melhor forma de validar as atividades e o fluxo de trabalho foi com o auxílio dos próprios autores e desenvolvedores do jogo tratado pelo *postmortem*. Esta etapa, portanto, possibilitou validar o método de extração e criação dos processos gráficos.

4.2 CONSTRUÇÃO DO SISTEMA DE RECOMENDAÇÃO

A segunda etapa (fase 2 da Figura 3), apresentada no Capítulo 6, consistiu na criação do sistema de recomendação. Primeiramente foi definido um contexto para o desenvolvimento de jogos, seguido da extração das características de contexto de cada *postmortem* coletado. Após, foi criado um algoritmo para comparar a similaridade dos projetos. Por fim, a parte da junção dos atributos de cada processo juntamente com as preferências do desenvolvedor, formando um novo processo customizado, que re-alimentará a ferramenta de similaridades.

Para a definição do contexto de projetos de jogos digitais, definiu-se características que determinam o tipo do projeto. Com o contexto definido, extraiu-se os dados de cada *postmortem* analisado, utilizando as informações filtradas na primeira etapa. Como explicado na seção 2.3, a forma como se desenvolve software depende muito do contexto do mesmo. Como não foi encontrado nada na literatura a respeito de um contexto de desenvolvimento de jogos digitais, houve a necessidade da concepção de um. O contexto foi criado baseando-se em toda a pesquisa sobre a área de jogos já realizada. Cada propriedade tem um motivo e função que afeta a forma de desenvolver o jogo, ou seja, o processo. O contexto criado para este trabalho não pretende ser definitivo, mas sim servir como base para novos estudos na sua refatoração e ampliação. Ainda, o sistema proposto foi estruturado de forma isolada, ou seja, a estrutura do contexto pode ser alterada e o mesmo continuará funcionando. Por fim, para popular o contexto criado, houve a necessidade de mais uma análise dos dados extraídos do *postmortem* e pesquisa em fontes externas.

A construção do sistema de recomendação teve duas sub-etapas: (1) encontrar as similaridades entre os projetos de jogos e (2) gerar um novo processo gráfico utilizando os elementos dos projetos de maior similaridade ao contexto de entrada.

O meio de encontrar a similaridade entre os projetos de jogos se deu utilizando uma técnica, denominada *PCA*, para achar correlações em múltiplas variáveis e reconhecer padrões (mais detalhes na Seção 2.5). Outras técnicas de aprendizado de máquina foram testadas, como utilizar um conjunto de treinamento e outro de teste juntamente com uma árvore de decisão para recomendar qual processo é mais apropriado dada uma entrada. No entanto, além de haver poucas amostras, algoritmos supervisionados necessitam de uma classificação para cada amostra (*label*). Uma

saída foi nomear cada processo resultante de cada amostra e então recomendar o tipo de processo mais semelhante aos parâmetros de entrada. Verificou-se então, que muitas informações foram perdidas além de ser uma contribuição muita fraca para o desenvolvedor, isto é, ajudaria muito pouco o time de desenvolvimento saber que o processo mais propício é adaptativo ou preditivo.

Outra tentativa foi utilizar algoritmos de agrupamentos como o *K-Means*, também fazendo uso de grupos de amostras de treinamento e teste. A recomendação do processo se daria a partir da junção dos processos das amostras do mesmo grupo (*cluster*). Nesse caso o problema foi a definição prévia do número de *clusters*. Não há como saber o número exato pois as amostras se mostraram bem espalhadas no plano, como visto no exemplo da plotagem do PCA (Figura 9).

A geração do processo parte da concatenação dos projetos similares, com ênfase nas características de contexto estabelecidas na entrada. O processo final gerado (recomendado) seguiu dois modelos padrões pré-definidos: **adaptativo**, focando em iterações rápidas e entregáveis completos; ou **preditivo**, com ênfase em um planejamento prévio maior. O sistema de recomendação foi o mais automatizado possível, com regras definindo qual modelo usar. No entanto, para as demais tarefas foi necessário o viés de um especialista.

4.3 VALIDAÇÃO

A validação (fase 2 da Figura 3) consistiu em duas etapas, sendo a primeira comparar a semelhança do **processo gerado** pela ferramenta de recomendação com relação ao **processo extraído** e validado através da análise do *postmortem*. Da seguinte forma:

1. Usar um dos processos extraídos como entrada para a ferramenta de recomendação. Para esta etapa foram utilizadas as características do projeto na forma de contexto.

2. Gerar um novo processo;

3. Calcular o *precision* e *recall* de cada elemento, dos processos gerados e extraídos, da seguinte forma:

    - Se o elemento do processo recomendado for compatível com o processo extraído, o mesmo foi considerado Verdadeiro Positivo (TP);
    - Se o elemento do processo recomendado não for compatível com o processo extraído, o mesmo foi considerado Falso Positivo (FP);

- Se algum elemento estiver no processo extraído, mas não apareceu no processo recomendado, então o mesmo foi considerado Falso Negativo (FN).

Utilizando a Tabela 2 como referência, utilizou-se as seguintes fórmulas, para o cálculo do *precision* e *recall*, respectivamente: $precision = TP/(TP + FP)$ e $recall = TP/(TP + FN)$.

Tabela 2 – Tabela usada para calcular a *precision* e *recall* dos elementos do processo.

Elementos do Processo	Recomendado	Não Recomendado
Extraído	Verdadeiro Positivo (TP)	Falso Negativo (FN)
Não Extraído	Falso Positivo (FP)	Verdadeiro Negativo (TN)

Fonte: elaborado pelo autor.

A segunda etapa da validação foi complementar à primeira, na qual entrou-se em contato com os autores dos projetos analisados e solicitou-se um retorno sobre o processo gerado pelo sistema de recomendação. O principal objetivo foi saber a viabilidade do novo processo gerado, por meio das seguintes perguntas:

1. “De forma geral, este fluxo de trabalho é similar ao que vocês utilizaram desenvolvendo o jogo?”
2. “Que elementos do processo não fazem sentido?”
3. “Você adicionaria algo importante a este fluxo?”
4. “Se vocês começassem um novo projeto de jogo, similar ao <JOGO PRÉVIO DO DESENVOLVEDOR>, qual seria a viabilidade de usar este novo processo?”

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou o método de desenvolvimento utilizado para atingir o objetivo proposto neste trabalho. O método foi dividido em três etapas, sendo a primeira a etapa de coleta, análise, construção e validação de processos de desenvolvimento de jogos a partir de *postmortems*. A segunda etapa foi a construção do sistema de recomendação de processos de desenvolvimento de jogos utilizando a técnica de análise de dados *PCA* para juntar os processos semelhantes e a construção de um contexto para projetos de jogos digitais. A última etapa foi a validação de alguns processos gerados pelo sistema de recomendação com o auxílio dos próprios autores/desenvolvedores dos projetos analisados.

5 ANÁLISE DE *POSTMORTEMS* E CONSTRUÇÃO DE PROCESSOS

Este capítulo apresenta o desenvolvimento da etapa de análise, extração, construção e validação dos processos de desenvolvimento a partir de *postmortems* escritos por desenvolvedores de jogos. A Seção 5.1 descreve a etapa de coleta dos *postmortems*. A Seção 5.2 demonstra como foi realizada a análise dos *postmortems*. A Seção 5.3 apresenta a forma em que os dados foram extraídos. A Seção 5.4 mostra como foram construídos os diagramas gráficos dos processos. A Seção 5.5 exibe o *feedback* dos autores dos *postmortems*. A Seção 5.6 resume os principais pontos do capítulo. Por fim, a Seção 5.7 discute os resultados deste capítulo.

5.1 COLETA DOS *POSTMORTEMS*

A coleta dos *postmortems* foi realizada de forma semi-automatizada, divida em 3 passos:

Passo 1: Utilizando técnicas de *web scraping*[1] e a ferramenta de linha de comando *wget*[2] os artigos usados na análise foram coletados da página sobre *postmortems* do *site Gamasutra* (*gamasutra.com/features/postmortem*). O *site*, além de ser uma referência para desenvolvedores de jogos também dedica uma seção inteira para artigos de relatos de experiências. Após, foi efetuado o *download* de todos os artigos e o mesmo foi transformado em arquivos no formato PDF utilizando a ferramenta *wkhtmlox*[3].

Além disso, foram filtrados e extraídos do artigo *postmortem* e da página do jogo na *Wikipédia* outras informações relativas ao jogo e armazenados em arquivo no formato *CSV* [4]. Os dados extraídos foram os seguintes:

- Dados extraídos da página do *postmortem*: *Game, Title, Author(s), Date, Url*.

- Dados extraídos da *Wikipedia*: *Mode(s), Genre(s), Platform(s), Engine, Developer(s), Publisher(s), Distributor(s), Releasedate(s), Designer(s), Programmer(s), Composer(s), Artist(s), Writer(s), Director(s), Producer(s)*.

[1] Técnica de extração de informações de *websites* (WIKIPEDIA, 2016c).
[2] Ferramenta de extração de arquivos via protocolos HTTP e FTP (WGET, 2017).
[3] Ferramenta de linha de comando que renderiza aquivos HTML em PDF (HTMLTOPDF, 2017).
[4] O tabela completa, com todas as informações, pode ser visualizada no *site* do projeto, no endereço: *https://polako.github.io/dissertation/pm-data.csv*.

Todo esse trabalho foi realizado por um *script*[5], escrito na linguagem *Python*, que iterava nas páginas e realizava o *download* e extração dos artigos.

O algoritmo funciona da seguinte forma: para cada *link* de artigo, o algoritmo, utilizando a estrutura de *tags* do *HTML*, busca por partes chaves do texto, como "título" e "nome do jogo". Esses dados são exportados para uma planilha (arquivo CSV). O *script* também faz uma busca na *Wikipedia* pelo artigo correspondente ao jogo, e as informações extras, contidas na caixa lateral, foram anexadas na planilha. Após, é feito o *download* completo da página, ou seja, do artigo *postmortem* no site *Gamasutra*. Por fim, é gerado o arquivo PDF e os arquivos locais são excluídos.

Passo 2: Com a lista dos dados de cada *postmortem*, foi verificado, com a leitura dos títulos, se os artigos realmente eram relatos de jogos ou de algum outro assunto[6]. O resultado final da filtragem foi uma lista totalizando 183 *postmortems*.

Por alguma razão, a página relativa a *postmortems* do site *Gamasutra* não continha artigos após o ano de 2014. Para tanto, foi realizado uma nova busca para integrar os dados restantes. O *script* foi modificado para fazer uma varredura nas 54 páginas resultantes da busca pelo termo *"postmortem"*, sendo que cada página continha 25 artigos, num total de 1350. No final, o número total de *postmortems* foi de **234, de 1997 até 2016**

Passo 3: Por fim, os artigos foram ordenados de forma descrescente, de acordo com a data de publicação. Isto foi feito para dar preferência aos *postmortems* mais novos, ou seja, projetos mais recentes.

5.2 LEITURA E ANÁLISE DOS *POSTMORTEMS*

Partindo do artigo mais recente, foi realizada a leitura dos *postmortems*, de forma manual. Na leitura, procurou-se **extrair todas as informações referentes ao processo utilizado durante o desenvolvimento do jogo**. Foi salientada toda e qualquer atividade mencionada pelo autor, características associadas ao time e ao projeto em si. Detalhes de design e características do jogo não foram levadas em consideração.

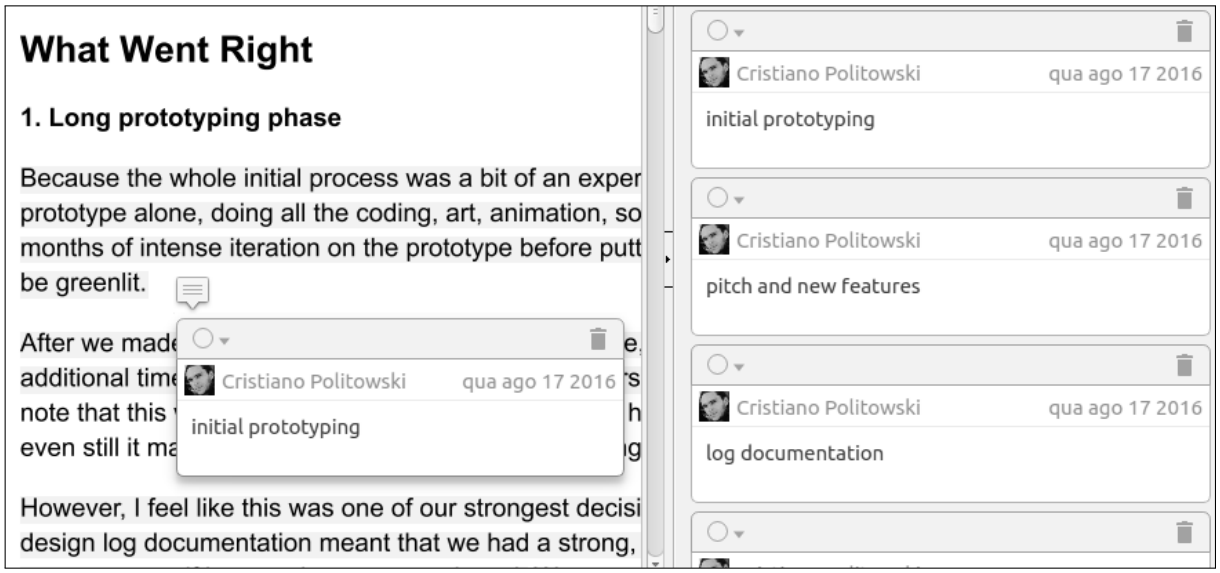
Para esta tarefa foi utilizada a ferramenta Mendeley[7] dada sua capacidade em criar destaques e anotações sobre os arquivos. Cada informação importante foi marcada e indexada com uma nota, conforme mostra a Figura 4, possibilitando uma rápida análise do projeto apenas olhando os índices. À direita da imagem podem ser visualizados os índices (notas) e na esquerda o texto em destaque.

[5] O código completo pode ser visualizado no *site* do projeto no seguinte endereço: *https://polako.github.io/dissertation/scripts/*

[6] Há *postmortems* sobre vários temas como música por exemplo.

[7] Ferramenta para gerenciamento de artigos: https://www.mendeley.com.

Figura 4 – Processo de leitura e análise dos *postmortems* utilizando a ferramenta *Mendeley*.



Fonte: elaborado pelo autor.

Alguns artigos analisados não continham informações relevantes à pesquisa. Este fato se dá pois esses relatos (*postmortems*) possuem estrutura livre na qual muitos autores, as vezes devido a restrições de privacidade, acabam relatando poucos detalhes ou atividades relacionadas ao design do jogo e a busca pela diversão. Nestes casos o artigo foi excluído e retirado da lista.

Um total de 100 *postmortems* foram analisados, resultando em **55 artigos** com informações suficientes para a extração do processo utilizado no desenvolvimento do jogo. A lista dos *postmortems* analisados está na Tabela 3.

5.3 ESTRUTURA E EXPORTAÇÃO DOS DADOS PARA FORMATO *MARKDOWN*

Com os dados das leituras dos *postmortems* foi possível separar as informações em diferentes grupos: **atividades**, **características**, **time** e ***feedback***. Transcreveu-se os dados relativos a cada um destes grupos em uma estrutura “chave:valor”, na qual a chave é o índice (ou nota) criado durante a leitura e o valor é a citação correspondente (texto em destaque).

Para facilitar a escrita e leitura do documento, optou-se pelo formato *markdown*[8], sendo os grupos definidos pelo título (“##”) e os índices e citações em uma estrutura de listas (“*”).

O exemplo da Listagem 1 mostra os elementos retirados da análise do *postmortem* do projeto do jogo *Slow down, bull*. A primeira linha define o nome do jogo. A linha 3 estabelece que, a partir daquele ponto, todo novo item faz parte da categoria

[8] *Markdown* é uma linguagem de marcação leve (WIKIPEDIA, 2016a).

Tabela 3 – Lista de artigos *postmortems* analisados.

#	Game and *postmortem* link	Date	Feedback?
1	The Banner Saga 2	June 10, 2016	
2	Stellaris	June 14, 2016	
3	Red Skies	May 18, 2016	yes
4	Ratchet & Clank (2016)	May 16, 2016	
5	Offworld Trading Company	June 7, 2016	yes
6	Mini Metro	June 15, 2016	yes
7	Vanishing Point	June 18, 2016	
8	Swing Racers	November 11, 2015	
9	Sunless Sea	March 4, 2015	
10	Slow Down, Bull	November 12, 2015	yes
11	Rpublique	April 12, 2016	
12	Race the Sun	February 1, 2016	
13	Prune	February 12, 2016	yes
14	Out There	April 2, 2015	yes
15	Ori and the Blind Forest	May 5, 2015	
16	NFL Rush Heroes & rivals	Decembe 9, 2015	yes
17	Never Alone	February 19, 2015	
18	Middle-earth: Shadow of Mordor	January 20, 2015	
19	Lost Within	August 26, 2015	
20	Lords of the Fallen	February 12, 2015	
21	INK	August 21, 2015	
22	I Can't Escape: Darkness	December 23, 2015	
23	Goat Simulator	February 20, 2015	
24	Far Cry 2	October 21, 2015	
25	Crashlands	March 7, 2016	
26	Consortium	February 4, 2016	
27	Cave Dash	Decembe 7, 2015	
28	Catlateral Damage	July 13, 2015	
29	Ashes of the Singularity	April 26, 2016	
30	80 Days	October 14, 2015	
31	Tropico 5	March 17, 2015	
32	Rollers of the Realm	December 31, 2014	
33	Jetpack High	September 25, 2014	
34	Dead State	March 11, 2015	
35	Real Boxing	January 21, 2013	
36	Natural Selection 2	February 26, 2013	
37	Kingdoms of Amalur: Reckoning	July 30, 2013	
38	Guacamelee!	September 23, 2013	
39	God of War: Ascension	June 19, 2013	
40	City Conquest	February 6, 2013	
41	Baldur's Gate: Enhanced Edition	April 15, 2013	
42	Anomaly Warzone Earth	January 28, 2013	
43	Zack Zero	July 3, 2012	
44	Sins of a Solar Empire: Rebellion	October 9, 2012	
45	Lumines Electronic Symphony	July 30, 2012	
46	Dust: An Elysian Tail	October 31, 2012	
47	Casey's Contraptions	June 22, 2011	
48	Trine	June 3, 2010	
49	The Path	July 22, 2010	
50	Scooby-Doo! First Frights	January 13, 2010	
51	Deadliest Warrior	November 11, 2010	
52	Brutal Legend	March 25, 2010	
53	Aaaaa! – A Reckless Disregard for Gravity	September 22, 2010	
54	Sins of a Solar Empire	April 28, 2008	
55	Spider-Man 2	June 21, 2013	

Fonte: elaborado pelo autor.

Listagem 1 – Exemplo da estrutura *markdown* utilizada para armazenar os dados oriundos da análise dos *postmortems*.

```
1  # Slow Down, Bull
2
3  ## team
4  * team
5    * Besides me, the team for Slow Down, Bull was composed entirely of contractors (though some
         still had some connections with Insomniac — a former animator gone freelance, a former
         intern of ours, a QA tester with game art aspirations).
6  (...)
7  ## activities
8  * **initial prototyping**
9    * Because the whole initial process was a bit of an experiment, we spent a long time with
       just me working on the prototype alone, doing all the coding, art, animation, sound,
       telemetry, and playtesting. It was roughly four months of intense iteration on the
       prototype before putting something together for a broad company playtest to be greenlit.
10 (...)
11 ## characteristics
12 * expertise source
13   * In a way, the act of consulting an expert became a form of delegation, and pure brain-
       expertise became a resource to be taken advantage of. It almost felt like being in an
       incubator for creating an indie game, with the process itself being relatively
       independent, but the fact that help on a complex programming struggle was just a desk or
        two away.
14 (...)
15 ## feedback
16 * I would switch the position of initial prototyping and early playtesting. The prototype had
     to exist before we could playtest afterall! It would also help to show that this part is
     an iterative cycle - it wasn't "we spent x time prototyping and x time playtesting," but
     rather many many loops of prototyping->playtesting->adjusting prototyping->more
     playtesting.
17 (...)
```

“team”. O mesmo vale para *“activities”* após a linha 7, *“characteristics”* após a linha 11 e *“feedback”* após a linha 15.

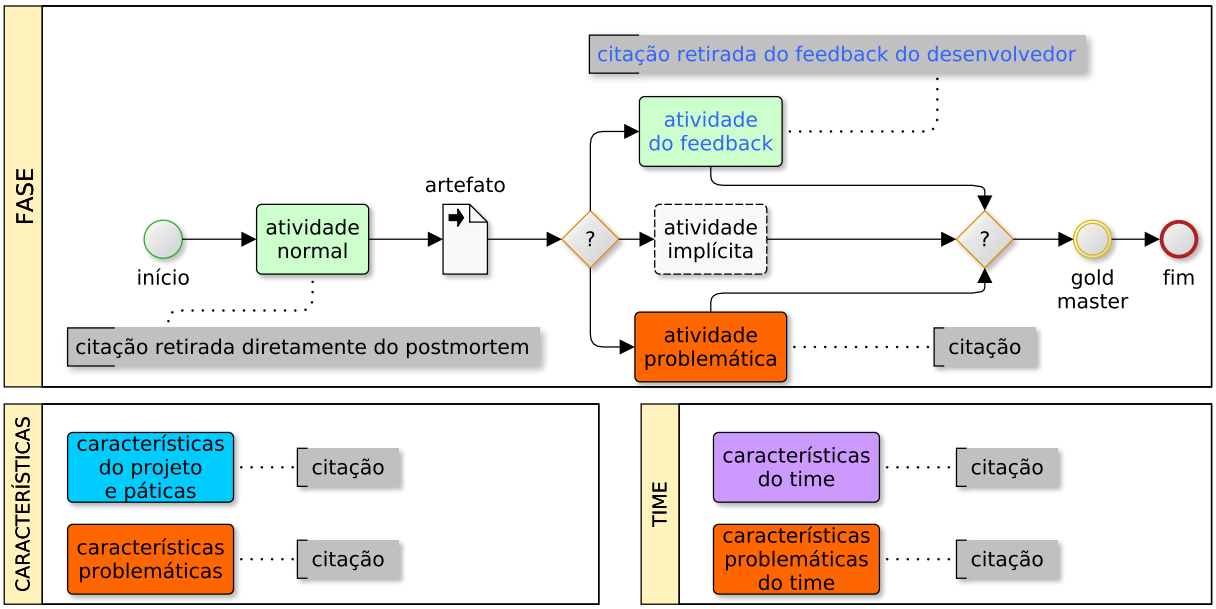
Os itens da listagem, definidos por começarem com asterisco (“*”), possuem dois níveis: o primeiro define o **índice** ou nome do item, como nas linhas 4 (*team*), 8 (*initial prototyping*) e 12 (*expertise source*); o segundo define a **citação** do item pai (índice), como nas linhas subsequentes 5, 9 e 13. Ainda, índices podem estar entre entre dois asteriscos (“** **”), como a atividade da linha 8. Isto representa que ocorreram dificuldades durante a execução daquela tarefa, ou seja, o desenvolvedor relatou algum problema durante a processo.

5.4 CRIAÇÃO DO DIAGRAMA GRÁFICO

Para melhor entender as atividades e fluxo seguido pelos desenvolvedores no desenvolvimento do jogo, uma visualização gráfica se faz necessária. Para tanto, foi definido um modelo padrão (ou *template*), do qual todos os processos descendem. Este modelo gráfico padrão foi criado a partir de estudos literários e pesquisas na área de desenvolvimento de jogos e pode ser visualizado na Figura 5.

Segundo Bates (2004), Moore e Novak (2010) e também constatado em trabalho prévio (POLITOWSKI *et al.*, 2016a), é uma prática comum entre os desenvolvedores,

Figura 5 – Modelo gráfico padrão, em BPMN, utilizado como *template* para criação dos processos.



Fonte: elaborado pelo autor.

embora não seja reconhecidamente um padrão, separar o desenvolvimento do jogo em três fases: **pré-produção**, **produção** e **pós-produção**. Na primeira fase normalmente há iterações na validação de ideias, testes de protótipos, *brainstorming*, criação do documento de design do jogo (*Game Design Document - GDD*), realização de *pitchs* com *publishers*, entre outras atividades. A segunda fase tende a começar somente quando a ideia central do jogo está definida, é também chamada de “fase de produção” ou *full production* e resulta no produto finalizado. A última fase consiste na aplicação de atualizações e *patches* para correção de problemas e melhorias.

Os **elementos** no processo podem ser de três tipos: **atividade** é o principal, é uma tarefa ou passo que descreve unidades de trabalho que resultem em um artefato. **Características e práticas** são padrões ou hábitos sistemáticos usados pelo time durante o ciclo de desenvolvimento. Por fim, **características do time** dizem respeito a qualquer detalhe, destacado pelo autor, sobre os atributos do time de desenvolvimento.

Cada elemento do processo pode ter três estados: **normal**, quando o desenvolvedor não relatou problemas durante a execução da atividade; **problemático**, quando o desenvolvedor encontrou algum tipo de problemas relacionado a atividade executada; e **feedback**, quando algum elemento é oriundo da validação após o *feedback* do desenvolvedor, e não extraído da análise do *postmortem*.

Elementos do processo podem conter **citações**, retiradas do *postmortem* e que validam a existência do elemento. A quase totalidade das atividades e características descritas nos processos possuem citações individuais para cada elemento. Dessa forma, juntamente com **citações de *feedback***, diminui-se o viés da etapa de análise e

extração dos elementos do processo.

Para desenvolver o modelo gráfico foi utilizada a ferramenta *GraphViz*[9], que permite criar diagramas em uma notação textual e então compilar para diversos formatos de arquivos como SVG, PNG e PDF. A sintaxe da linguagem específica de domínio do *GraphViz* é simples e, com poucas linhas de código, o modelo padrão dos processos foi construído, como mostra o código na Listagem 2. Por exemplo, a linha 1 especifica que o desenho vai ser um dígrafo. As linhas seguintes mostram algumas configurações do mesmo, como sentido (esquerda-pra-direita na linha 2), tipo de fonte (*arial* na linha 3), etc. A partir destes elementos são definidos cinco sub-grafos, cada um com seus respectivos atributos: *team*, na linha 10, que vai conter os elementos do processo relacionados ao time de desenvolvimento. *Characteristics*, linha 15, que diz respeito às características e práticas utilizadas durante o desenvolvimento. E as três fases, *pre-production*, linha 19, contendo elementos da pré-produção; *production*, linha 29, com elementos da fase de produção e *post-production*, linha 44, com elementos da pós-produção.

A tarefa de extrair os elementos do processo do arquivo *markdown* para o arquivo no formato *GraphViz* (*.dot*) foi realizada utilizando um *script* que percorria todo o arquivo e inseria os elementos, já com seus atributos gráficos, no modelo padrão previamente definido. Após esta fase, ainda restava fazer uma última análise e ligar elementos, agora já na sintaxe do *GraphViz*. Esta etapa normalmente era a mais rápida, visto que o *script* já deixava o arquivo praticamente completo. O algoritmo[10], em linguagem *Python*, tem seu funcionamento, de forma geral: ler o arquivo *markdown* (*.md*), iterar no texto buscando palavras chaves (índices), criar os elementos e inseri-los no modelo padrão previamente citado. Ao final, gerar um novo arquivo, de extensão *.dot*, pronto para ser compilado e transformado em arquivo gráfico como PNG, SVG ou PDF.

Na Figura 6 é mostrado o processo extraído e validado pelo autor (ver Seção 5.5) do *postmortem* do jogo *Slow down, Bull*[11]. O processo tem duas fases: pré-produção e produção, não contendo a fase de pós-produção. Cada fase do processo tem um início e um fim, demarcado pela pequena esfera escura. Em ambas as fases, os elementos da cor verde são as atividades que ocorreram de forma normal durante o desenvolvimento enquanto que as vermelhas encontraram percalços. Na maioria das vezes não foi possível descrever completamente o fluxo das atividades no processo e, nestes casos, os elementos foram alocados após um ponto de interrogação (“?”). Na produção, o elemento de nome *“gold”* indica o software pronto, ou seja, o jogo completo. Cada

[9] Ferramenta de visualização de grafos: http://www.graphviz.org

[10] Disponível no endereço: *https://polako.github.io/dissertation/scripts*.

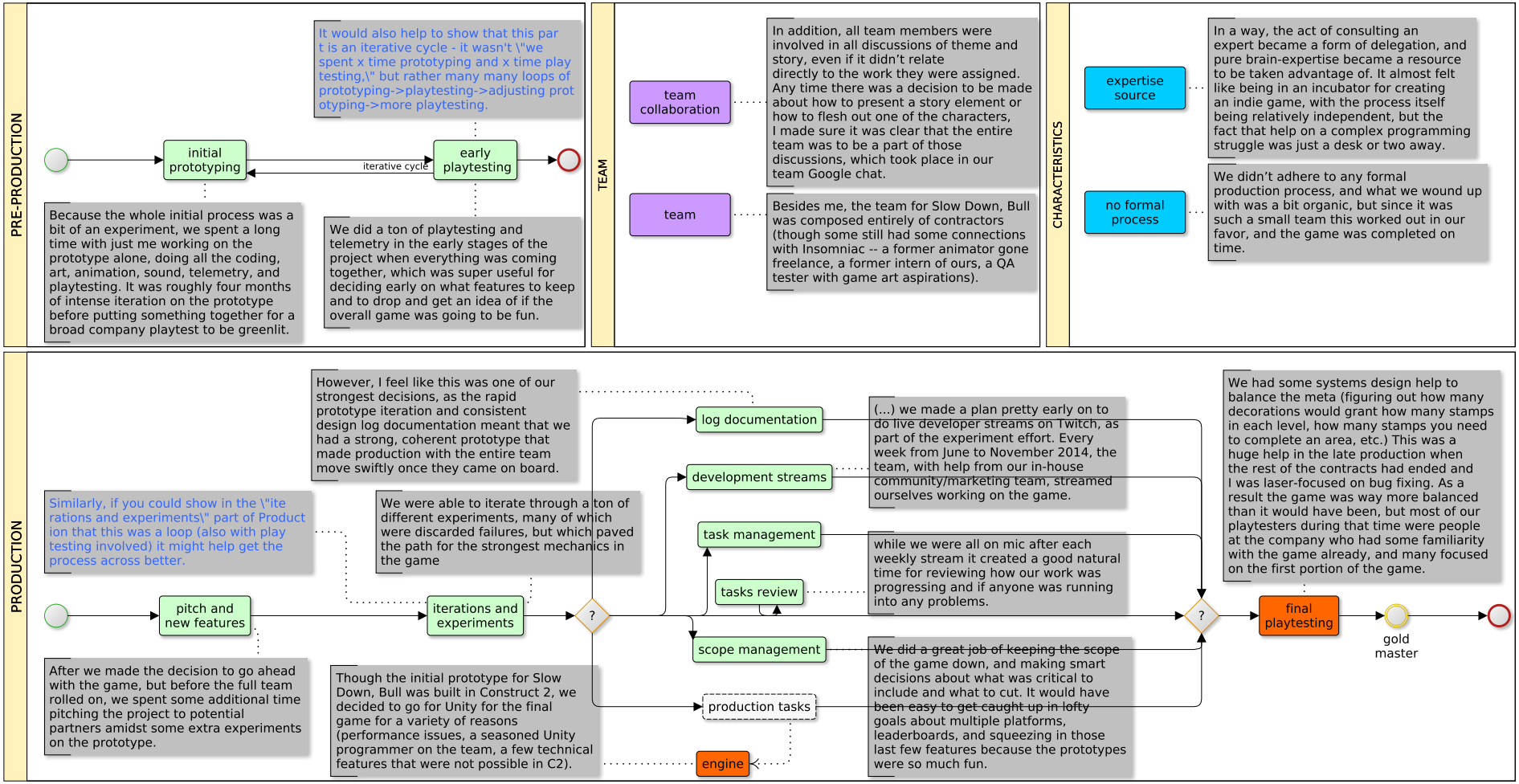
[11] Para evitar consumo de espaço no trabalho, apenas um exemplo de processo extraído é mostrado. Todos os outros 54 processos encontram-se no endereço: *https://polako.github.io/dissertation/projects*.

Listagem 2 – Trecho do template gráfico criado mostrando a sintaxe da linguagem *DOT*.

```
/*DOCs: http://www.graphviz.org/doc/info/attrs.html*/
digraph G {
  rankdir=LR;
  fontname="arial";
  compound=true;
  center=true;
  bgcolor=antiquewhite;
  edge [color=gray];
  node [style=filled, shape=box, fontname=arial, width=2];
  /*team*/
  subgraph cluster_team {
    node [fillcolor="darkorchid1"]
    label="Team"
  }
  /*characteristics*/
  subgraph cluster_char {
    node [fillcolor="deepskyblue1"]
    label="Characteristics & Practices"
  }
  subgraph cluster_pre {
    bgcolor=white;
    style=bold;
    start1 [shape=point, width=.2];
    end1 [shape=point, width=.2];
    node [fillcolor="aquamarine3"]
    start1 -> {} -> end1
    label="Pre-Production";
  }
  subgraph cluster_pro {
    bgcolor=white;
    style=bold;
    start2 [shape=point, width=.2];
    end2 [shape=point, width=.2, label="gold master"];
    gm [shape=circle, fillcolor="gold", fixedsize=true, width=.5, style="filled,dashed", label
        ="gold"]
    t1 [shape=diamond, label="?", width=1]
    t2 [shape=diamond, label="?", width=1]
    pt [label="production tasks",style=dashed]
    node [fillcolor="aquamarine3"]
    start2 -> t1 -> {pt} -> t2 -> gm -> end2
    label="Production";
  }
  subgraph cluster_pos {
    bgcolor=white;
    style=bold;
    start3 [shape=point, width=.2];
    end3 [shape=point, width=.2];
    node [fillcolor="aquamarine3"]
    start3 -> end3
    label="Post-Production";
  }
  end1 -> start2 [label="", style=dashed];
  end2 -> start3 [label="", style=dashed];
}
```

atividade possui uma citação direta das palavras do autor do *postmortem*, no bloco cinza. Os elementos com texto azul denotam a refatoração após a validação do autor. Por fim, há dois quadros contendo as características e práticas gerais e características to time.

Figura 6 – Processo de desenvolvimento do jogo *Slow down bull*.



Fonte: elaborado pelo autor.

5.5 *FEEDBACK* DOS AUTORES DOS *POSTMORTEMS*

Para aumentar a precisão dos processos extraídos e diminuir o viés da análise dos *postmortems*, foi adicionada uma fase de coleta de *feedback* dos autores dos artigos. Nesta fase, foi feito contato com alguns dos desenvolvedores/autores para avaliar o nível de precisão dos processos extraídos. Houveram 7 respostas dos 20 autores requisitados, dos seguintes jogos: *"Red Skies", "Offworld Trading Company", "Mini Metro", "Slow Down, Bull", "Prune", "Out There", "NFL Rush Heroes & rivals"*. Foi solicitado que os autores analisassem o processo extraído em modo gráfico, e respondessem três questões (mais detalhes no fomulário completo no Apêndice A):

- "De forma geral, este fluxo de trabalho é similar ao que vocês utilizaram desenvolvendo o jogo?"
- "Que elementos não fazem sentido?"
- "Você adicionaria algo importante?"

5.5.1 *Feedback* do processo do jogo *"Red Skies"*

O autor do *postmortem* do jogo *"Red Skies"*[12] chamou a atenção para dois detalhes que não refletiam o processo utilizado pelo time no projeto: a posição do elemento *deadline* (item a-1) e a retirada da fase de pós-produção (item a-2). Por fim, apesar de não conhecer a sintaxe utilizada na criação do gráfico, o autor entendeu o seu significado, destacando a acurácia do processo extraído (item a-3).

(item a-1) *"There is a major inaccuracy in the deadline position: We only set the final deadline 2 months before the release (5 months in)."*

(item a-2) *"Also, we didn't do any post-production, since the game wasn't really a hit (only a few marketing attempts)."*

(item a-3) *"Appart from that, I'm not totally sure I understand how to read the graphic, but most of it seems pretty accurate. I guess the red bricks describe the steps we failed to do correctly, and the greens the ones that went more smoothly."*

5.5.2 *Feedback* do processo do jogo *"Offworld Trading Company"*

O autor do *postmortem* do jogo *"Offworld Trading Company"*[13] considerou o processo extraído bem preciso, com algumas ressalvas: o programa de *funding* aconteceu antes do *early access* (item b-1) ; o desenvolvimento iterativo ocorreu somente na

[12] *https://polako.github.io/dissertation/projects/3_red_skies.html*

primeira etapa do desenvolvimento, depois seguiu-se um plano, típico de processos híbridos (junção de processos iterativos e preditivos (POLITOWSKI *et al.*, 2016a)) (item b-2); assim como as iterações, o *feedback* da comunidade foi utilizado somente no início do projeto (item b-3).

(item b-1) *"(...) the Founders Program happened before we went up on Early Access, about 6 months earlier."*.

(item b-2) *"Daily Iteration is sort of what we were doing before we went up on Early Access. After that, we usually had a schedule so we knew whether the next update was 1, 2, or 3 weeks away and could adjust our development accordingly."*

(item b-3) *"Feedback from the community was most useful during the first week after an update came out because then they were closest to our version. As we moved forward, we had to rely on the next_version branch for feedback from our more hardcore players."*

5.5.3 *Feedback* do processo do jogo *"Mini Metro"*

O autor do *postmortem* do jogo *"Mini Metro"*[14] foi sucinto ao dizer que o processo "pareceu-lhe" correto (item c-1). No entanto, não foi capaz de contribuir com mais detalhes.

(item c-1) *"That looks about right to me, but I haven't worked with diagrams like that since UML at university 15 years ago so I can't say I understand exactly what it's trying to say."*

5.5.4 *Feedback* do processo do jogo *"Slow Down, Bull"*

A autora do *postmortem* do jogo *"Slow Down, Bull"*[15] apontou vários detalhes a serem refatorados no processo: primeiramente a inversão de posições entre os elementos *prototyping* e *early playtesting* além de frisar que este processo era iterativo (item d-1). De forma semelhante, a autora sugeriu mostrar um *loop* entre as atividades *iterations and experiments* e *playtesting* (item d-2); por fim, corrigiu um engano na fase de extração, na qual havia um elemento de *deadline "greenlight"*, que na verdade não se referia ao processo de aceitação da publicadora *Valve* (item d-3).

[13] *https://polako.github.io/dissertation/projects/5_offworld_trading_company.html*
[14] *https://polako.github.io/dissertation/projects/6_mini_metro.html*
[15] *https://polako.github.io/dissertation/projects/10_slow_down_bull.html*

(item d-1) *“I would switch the position of initial prototyping and early playtesting. The prototype had to exist before we could playtest afterall! It would also help to show that this part is an iterative cycle (...)”*

(item d-2) *“Similarly, if you could show in the "iterations and experiments"part of Production that this was a loop (also with playtesting involved) it might help get the process across better.”*

(item d-3) *“I’m a little confused about the "greenlight" box under production (...) Are you perhaps referring to Steam Greenlight? If so, we did not go through the steam greenlight process due to the established studio relationship with Valve before Steam Greenlight became a thing.”*

5.5.5 *Feedback* do processo do jogo *“Prune”*

O autor do *postmortem* do jogo *“Prune”*[16] também concordou com a forma geral em que o processo foi mostrado, adicionando dois detalhes: juntou as atividades de prototipagem, priorização de tarefas e teste na fase de pré-produção (item e-1). Por fim, ressaltou o fluxo iterativo entre o planejamento, priorização de tarefas e teste (item e-2).

(item e-1) *“Yeah, that’s about right. I would maybe call the first 6 months of work on the game "pre-production,"which included things like prototyping, task prioritization, and playtesting.”*

(item e-2) *“I would say the key part of my development process was the iterative loop of plan/prioritize-prototype-playtest.”*

5.5.6 *Feedback* do processo do jogo *“Out There”*

O autor do *postmortem* do jogo *“Out There”*[17] foi sucinto mas de forma positiva em relação a análise e processo extraído: ressaltou a precisão da análise (item f-1); adicionou um *vertical slice* antes do início da produção (item f-2).

(item f-1) *“Your analysis is pretty much accurate (...)”*

(item f-2) *“I’ve made a teaser trailer before starting production (...)”*

5.5.7 *Feedback* do processo do jogo *“NFL Rush Heroes & rivals”*

O autor do *postmortem* do jogo *“NFL Rush Heroes & rivals”*[18] foi prestativo em sua análise, inclusive desenhando um esboço do que foi o fluxo de atividades durante

[16] *https://polako.github.io/dissertation/projects/13_prune.html*
[17] *https://polako.github.io/dissertation/projects/14_out_there.html*

a produção do jogo. Ainda, esclareceu algumas dúvidas que restaram em um segundo contato por *e-mail*. Algumas das principais considerações do autor foram: modificar a posição das tarefas que estavam em produção para a fase de pré-produção (item g-1); ressaltou que devido aos protótipos houve pouca iteração na fase de produção (item g-2); explicou que também faz uso de processos híbridos de desenvolvimento (item g-3).

(item g-1) *"Production tasks were assigned before production actually began. (I'd move it over to the left one space)"*

(item g-2) *"Typically thanks to the prototypes we didn't need to iterate much after production began. This was a bit of a blessing and a curse because we were able to hit our very short deadlines but some of the games suffered for it. If the game was passed to a different designer the new designer had a chance to iterate on the project a little."*

(item g-3) *"If you're looking into methodologies the one we used mostly was the waterfall methodology for the majority of my employment. Towards the end we started moving into agile."*

5.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a primeira etapa da construção do sistema de recomendação de processos para jogos digitais. Nesta etapa consta a coleta e análise dos *postmortems* dos jogos. Além da extração do processo de desenvolvimento e a validação do mesmo juntamente com os desenvolvedores/autores.

Um total de **100 *postmortems* foram analisados** resultando em **55 processos extraídos**, sendo **7 destes validados e refatorados** juntamente com os desenvolvedores.

O processo de análise foi feito manualmente. Entretanto as demais tarefas foram automatizadas quase que em sua totalidade, utilizando *scripts* isolados na linguagem *Python* e ferramentas integradas.

A geração dos gráficos dos processos extraídos foi realizada de forma semi-automatizada, também com o auxílio de um *script*, que "lia" o arquivo de elementos extraído da análise dos *postmortems* e gerava um novo arquivo na sintaxe *GraphViz*, no qual pode ser compilado para um arquivo PNG ou SVG ou PDF[19].

[18] *https://polako.github.io/dissertation/projects/16_nfl_rush.html*

[19] Todos os processos extraídos encontram-se no site do projeto, no endereço: *https://polako.github.io/dissertation//projects/*

5.7 DISCUSSÃO DOS RESULTADOS DO CAPÍTULO

O uso de *postmortems* para análise de projetos de jogos, embora não seja novidade, tem neste trabalho outro objetivo: o intuito de extrair o processo de software utilizado pelos desenvolvedores. Mesmo que a intenção dos relatos não seja esta, ainda assim é possível extrair informações relevantes ao processo de desenvolvimento e, em muitos casos, montar um fluxo das atividades realizadas pelo time.

Algo que resolveria alguns problemas da análise e ajudaria a comunidade de desenvolvedores seria a criação de um **repositório comum de experiências**. Neste repositório seriam depositados os relatos de projetos já finalizados e, em uma estrutura mais rígida, descritos os erros e acertos durante o projeto.

Esta fase da construção do sistema de recomendação resultou em 55 processos gráficos, extraídos da análise de *postmortems*. Pode-se dizer, então, que os diagramas gráficos são uma representação do que ocorreu durante a fase de desenvolvimento do jogo. Assim sendo, é uma forma rápida de consulta dos fatos ocorridos. Do ponto de vista da engenharia de software, não considerando detalhes de *gameplay* e design, é mais fácil realizar a leitura de um fluxo de atividade em formato gráfico do que um texto longo. Desta forma, **os processos extraídos e o sistema de extração podem servir como uma ferramenta de análise de projetos passados**. Por exemplo, uma empresa que tenha desenvolvido 10 jogos e que pretende fazer uma retrospectiva de todos eles, pode utilizar a ferramenta de extração para gerar 10 diagramas gráficos e então fazer comparações, agrupar projetos semelhantes e até definir métricas e medir a evolução da produtividade do time de desenvolvimento.

6 CONSTRUÇÃO DO SISTEMA DE RECOMENDAÇÃO

Este capítulo apresenta a construção do algoritmo de recomendação. A Figura 7 detalha a visão geral do sistema de recomendação. Inicialmente o sistema recebe como entrada o contexto do projeto, definido como atributos de contexto e características do time de desenvolvimento. Estes dados são capturados utilizando um formulário, preenchido pelos desenvolvedores ou engenheiro responsável pelo projeto. Os dados resultantes são armazenados em um arquivo *.CSV*, juntamente com os dados de contexto de outros projetos, no qual cada linha representa um projeto.

O "*script* de similaridades" recebe este arquivo, juntamente com as características do time, e realiza o cálculo *PCA*, gerando o *biplot* das amostras, juntamente com uma lista dos projetos mais similares ao contexto de entrada.

O "*script* de recomendação" recebe esta lista como entrada e realiza três ações: (1) concatena os arquivos *markdown* dos projetos mais similares, contendo os elementos extraídos das análises dos *postmortems*; (2) abstrai a descrição dos elementos do processo utilizando o modelo de abstração; (3) utiliza os modelos padrões de processos (preditivo ou adaptativo) como base para gerar um novo processo recomendado.

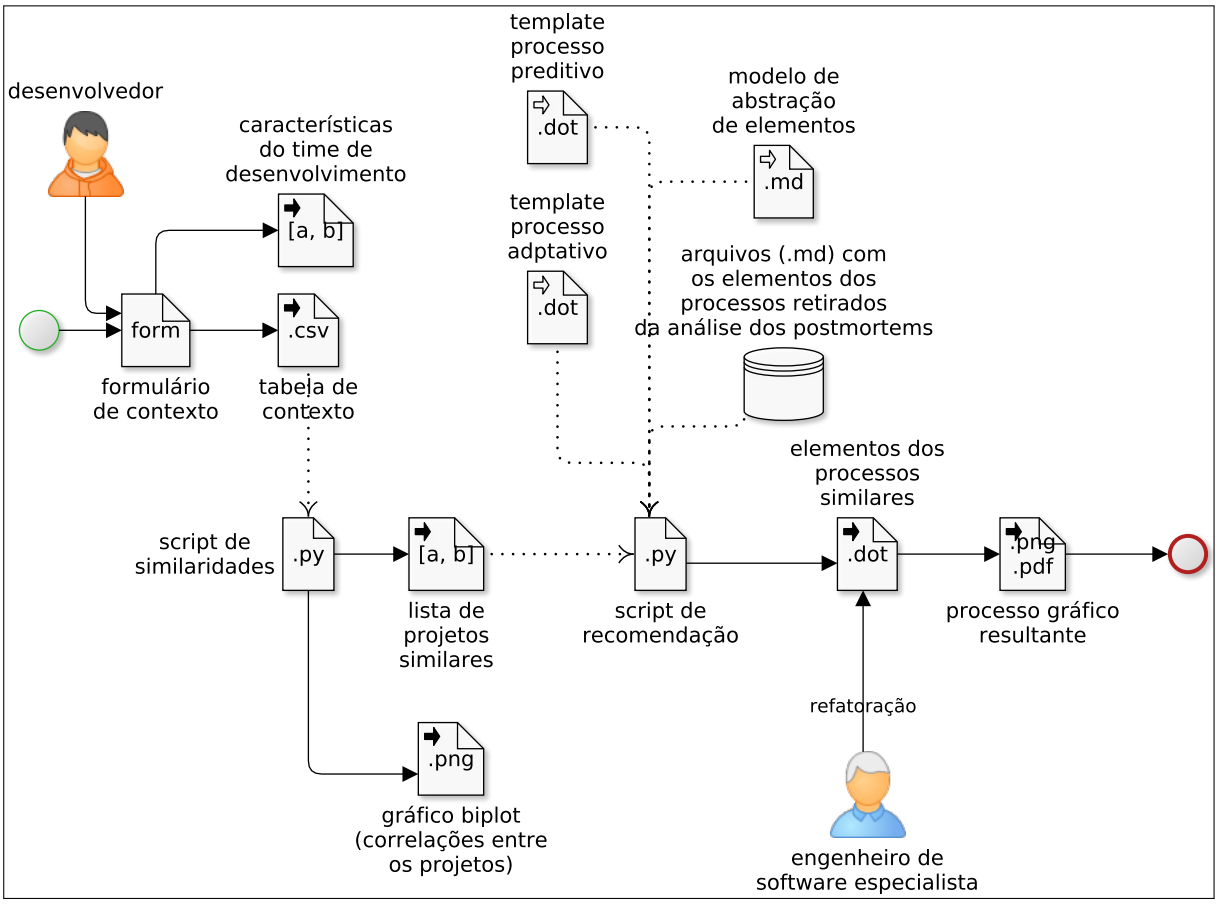
O arquivo gerado (*.dot*) é refatorado manualmente por um especialista, removendo alguma eventual inconsistência no processo. Por fim, o arquivo é compilado para o formato gráfico desejado.

Este capítulo está divido da seguinte forma. A Seção 6.1 descreve a criação do contexto para projetos de jogos digitais. A Seção 6.2 mostra a criação do algoritmo de similaridades, que utiliza o contexto para ilustrar os projetos análogos. A Seção 6.3 detalha a forma com que os processos similares foram associados formando o processo recomendado. A Seção 6.4 discute o processo resultante. A Seção 6.5 resume os principais pontos do capítulo. Por fim, a Seção 6.6 discute os resultados deste capítulo.

6.1 DEFINIÇÃO DO CONTEXTO DE PROJETOS DE JOGOS

Apesar das abordagens de contextualização de software, detalhadas na seção 2.3, representarem um ponto de partida para definir as características de um projeto de software, contextualizar um jogo envolve mais variáveis quando comparado a um projeto tradicional de software. Isto se dá devido a maior multidisciplinaridade encontrada em times de desenvolvimento de jogos, ou seja, a gama de competências dos profissionais envolvidos é maior[1]. Por conseguinte, definir um contexto que contemple todas as características do processo de desenvolvimento de um jogo digital não é tarefa trivial. Para este projeto, utilizando dos pareceres dos desenvolvedores

Figura 7 – Visão geral do Sistema de recomendação.



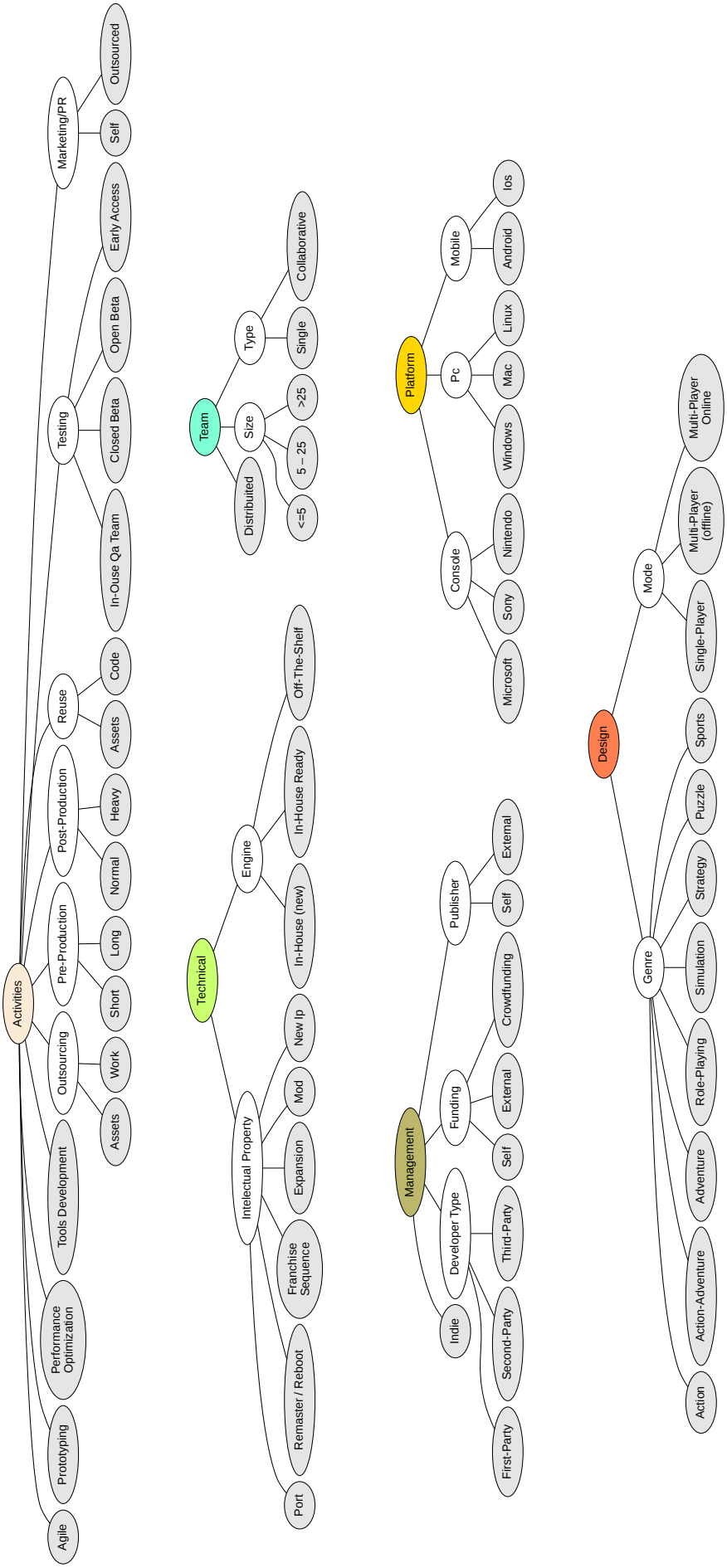
Fonte: elaborado pelo autor.

(POLITOWSKI *et al.*, 2016b), as análises dos *postmortems* (POLITOWSKI *et al.*, 2016a) e demais pesquisas (capítulo 3) entendeu-se que o **contexto do jogo deve conter propriedades que estejam relacionadas, de alguma forma, na maneira com que jogo vai ser desenvolvido, ou seja, o processo**. Para tanto, este trabalho propõe um conjunto de variáveis que, quando definidas, **caracterizam o tipo de projeto de jogo digital**.

O contexto foi dividido em seis grupos, descritos na Figura 8 e detalhados nas seções seguintes: contexto em nível de **atividades**, **time**, **gerenciamento**, **técnico**, **plataforma** e **design**. Cada elemento descrito possui a porcentagem correspondente, entre colchetes (“[ ]”), oriundas da análise das 55 amostras (projetos de jogos), ou seja, a porcentagem de projetos que possuíam aquela característica. Na tabela do Apêndice D são mostrados os dados extraídos de cada um dos 55 *postmortems* analisados.

[1] Evidentemente vai depender da complexidade do projeto envolvido e do tipo de jogo a ser criado. Ou seja, o projeto de um jogo simples vai demandar menor atribuições do que um projeto de um software de maior escala.

Figura 8 – Elementos do contexto para projetos de jogos.



Fonte: elaborado pelo autor.

6.1.1 Contexto em nível de Atividades

Neste nível de contexto, são detalhados os elementos, atividades e práticas do processo de desenvolvimento do jogo:

- ***Agile*** [32,73%]: É comum na comunidade de desenvolvedores de jogos a adoção de práticas ágeis mesmo que de forma inconsciente (PETRILLO; PIMENTA, 2010). É dada esta classificação para todo projeto que, independente do modelo (adaptativo, preditivo, etc), adota práticas ágeis de desenvolvimento de software.

- ***Prototyping* (prototipagem)** [66,67%]: Prototipagem é etapa na qual o time cria protótipos para testar uma ideia, ou seja, desenvolve um jogo incompleto (sem acabamento), mas funcional. Essa é uma prática corriqueira no desenvolvimento de jogos em que iterações são realizadas em busca do fator "diversão".

- ***Performance Optimization* (otimização de desempenho)** [16,67%]: O sucesso de um jogo não depende apenas de belos gráficos mas também da fluidez de sua jogabilidade. Para que isto aconteça é necessário que a taxa de atualização da tela por segundos (*frames per second*) seja alta e constante. Esta não é uma tarefa fácil e muitos desenvolvedores, preocupados com este detalhe, destinam uma etapa ou porção do tempo de desenvolvimento para realizar estes ajustes e otimizações no desempenho. Esta tarefa é ainda mais complicada quando o jogo é multiplataforma, visto as diferentes arquiteturas dos sistemas bem como a vasta gama de placas gráficas.

- ***Tool Development* (desenvolvimento de ferramentas)** [27,78%]: Desenvolver um jogo requer uma equipe multidisciplinar com profissionais que variam de especialistas técnicos a artistas. Para que o desenvolvimento seja mais fluído e não tenha gargalos, algumas ferramentas são necessárias para que artistas, designers e qualquer outro que não tenha afinidade com o código fonte possam agregar ao jogo ou testar uma funcionalidade sem depender de outra pessoa da equipe. É uma prática comum a criação de *engines* próprias, em oposição ao uso de uma solução pronta de terceiros.

- ***Outsourcing - Assets/Work* (terceirização de componentes ou mão-de-obra)** [*assets* 20,37% / *work* 36,36%]: A prática de contratação de terceiros (*outsourcing*) é muito utilizada, principalmente em times pequenos e desenvolvedores *indies*. Para isso, equipes contratam o auxílio de profissionais externos para agilizar o desenvolvimento e criação de componentes. Essa prática acontece também em períodos de forte *crunch* quando o *deadline* está próximo. É comum também a compra de componentes prontos em lojas especializadas, principalmente modelos 3D.

- ***Pre-Production* (pré-produção)** [*short* 64,81% / *long* 36,36%]: a fase de pré-produção acontece, notoriamente, antes da fase de produção e contém tarefas relativas a concepção do jogo, da discussão de funcionalidades, prototipagem, testes, levantamento de requisitos, análise de *feedbacks*, documentação, elaboração do documento de design, entre outras. Nesta fase não se trabalha no jogo em si, mas sim na base que sustentará a fase seguinte, de produção. A etapa de pré-produção também é utilizada para construção de ferramentas e *engines*. Alguns times, principalmente *indies* ou com prazo muito curto, preferem pular esta fase e entrar direto em produção.

- ***Post-Production* (pós-produção)** [*normal* 76,36% / *heavy* 18,52%]: a fase de pós-produção, evidentemente, se dá após a produção, ou seja, quando o jogo foi publicado. Esta fase é caracterizada por entregas de *patches* de correção, atualizações de balanceamento e adição de conteúdo ao jogo. Uma pos-produção **pesada (*heavy*)** indica que o jogo foi lançado de forma equivocada, contendo muitos problemas e, por esta razão, a necessidade da adição de muitos pacotes com correções.

- ***Reuse* (reúso)** [*code* 14,81% / *assets* 14,81%]: o reúso não é tão popular no desenvolvimento de jogos quando comparado ao desenvolvimento de software tradicional. Muitas vezes jogos são mal vistos pela comunidade caso algum componente seja reconhecido de outro jogo[2].

- ***Testing* (teste)**: teste é uma etapa vital e indispensável para o sucesso de um jogo. É correto afirmar que, quanto mais bem testado um jogo foi, mais polido ele será, sendo que o mesmo vale para software tradicionais. A área de testes é enorme e existem várias formas de testar e receber *feedback* de um jogo, separadas aqui em quatro:

  - ***In-House QA Team*** [59,26 %]: o time possui uma equipe que faz a garantia da qualidade e retorna *feedbacks* para o time de desenvolvedores. Esta é a alternativa mais custosa e normalmente apenas grandes desenvolvedoras possuem.
  - ***Closed Beta (user)*** [38,89 %]: testes fechados com usuários são muito utilizados e aproximam o desenvolvedor de seu público. Detalhe importante é que o retorno dos jogadores, apesar de não ser tão técnico e preciso, possui outro viés, que valida a jogabilidade.
  - ***Open Beta (user)*** [9,26 %]: funciona da mesma forma que o beta fechado, porém com um público aberto.

[2] Fato semelhante aconteceu na sequência de *"Dragon Age"*, *"Dragon Age 2"*, cujo prazo de desenvolvimento era muito curto (14 meses) e muitos componentes e lugares do jogo foram reutilizados.

  - ***Early Access (user)*** [9,26 %]: atualmente é uma prática bem corriqueira vender o jogo de forma antecipada e, com isso, também entregá-lo antecipadamente. Nesse hiato de tempo até o anúncio global é possível usar o *feedback* para realizar alguns ajustes.

- ***Marketing/Pr - Self/Outsourced*** **(marketing)** [*self* 81,82% / *outsourcing* 5,56%]: o sucesso ou fracasso de um jogo muitas vezes depende de quanto foi investido no marketing do mesmo, visto que há uma vasta quantidade jogos sendo lançados todos os dias[3] e conquistar uma posição de destaque nesse mundo passa por uma boa propaganda. Desenvolvedores *indies*, devido ao orçamento limitado, utilizam uma estratégia mais próxima do usuário ou tentam atingir especialistas em *reviews* formadores de opinião.

6.1.2 Contexto em nível de Time

Kruchten (2013) define o contexto do tamanho do "time" como pequeno, médio ou grande. Apesar de informativo, esse atributo é muito vago. Há vários detalhes que norteiam o time de desenvolvimento possuindo papel importante no processo de construção do jogo. Ao desenvolver jogos, é comum terceirizar componentes para que não seja necessário incorporar um profissional com aquela expertise ao time. Isso, portanto, influi diretamente no *pipeline* de produção e no processo de desenvolvimento.

Outro detalhe é a profundidade de conhecimento dos profissionais no time. Como há muita multidisciplinaridade no ambiente de desenvolvimento, ou seja, profissionais de várias áreas distintas (computação, design, músicos, escritores, entre outros) é comum ter times do tipo *cross-functional*[4]. No entanto, também é possível ter times com habilidades mais variadas chamados de profissionais *T-shaped skills*[5]. Essas variações afetam diretamente o *pipeline* de produção de componentes, por exemplo.

Isto posto, foram definidos três atributos para este contexto: o tamanho do time, o tipo do time e se o mesmo é distribuído:

- ***Team Size*** **(tamanho do time)** [*small* 44,23% | *medium* 27,45% | *big* 29,41%]: Para um time de desenvolvimento de jogos o tamanho do time foi divido em três: pequeno (**Small**), até 5 desenvolvedores; médio (**Medium**), de 5 a 25 desenvolvedores; e grande (**Big**), com mais de 25 desenvolvedores.

[3] Por exemplo, apenas na plataforma Steam (PC), em outrobro de 2016 foram lançados mais de 450 jogos.

[4] Time com diferentes expertises trabalhando em um único objetivo: https://en.wikipedia.org/wiki/Cross-functional_team.

[5] Metáfora que descreve pessoas com diversas expertises e menos profundidade em apenas um campo de conhecimento: https://en.wikipedia.org/wiki/T-shaped_skills

- ***Team Type* (tipo do time)** [*single* 67,27% | *collaborative* 22,22%]: O tipo do time foi separado em *single* (único), composto de apenas uma equipe, e *collaborative* (colaborativo), composto de duas ou mais equipes trabalhando para o mesmo fim.

- ***Distributed* (distribuído)** [41,82%]: O time é definido como distribuído quando os profissionais envolvidos não trabalham no mesmo espaço, normalmente fazendo *home-office*.

6.1.3 Contexto em nível de Gerenciamento

Este nível de contexto se refere a detalhes gerenciais do projeto, características da empresa e do time e desenvolvedores:

- ***Developer Type* (tipo da desenvolvedora)**: Na indústria de jogos, *Developer* ou Desenvolvedora é a denominação da empresa/estúdio ou pessoa responsável pela criação e desenvolvimento do jogo (BETHKE, 2003; MCGUIRE; JENKINS, 2008). Algumas delas são financiadas por *publishers* e/ou recebem apoio nas atividades de marketing (BATES, 2004). A divisão entre desenvolvedoras e publicadores nem sempre é clara, pois algumas *publishers* também são *developers*, como *Electronic Arts's EA Canada*, *Square Enix's studios*, *Activision's Radical Entertainment*, *Nintendo EAD*, *Sony's Polyphony Digital* e *Naughty Dog*. Desenvolvedoras normalmente são classificadas conforme sua relação com as fabricantes, *publishers*, meio de distribuição do jogo, liberdade criativa entre outros detalhes:

  - ***First-Party*** [3,70%]: São consideradas *first-party developers* estúdios desenvolvedores que fazem parte da empresa que fabrica o hardware e, por esta razão, seus jogos são exclusividades para aquela plataforma específica.
  - ***Second-Party*** [7,41%]: *Second-party developers* são estúdios que possuem contrato com a fabricante para desenvolver exclusivamente para o respectivo hardware ou estúdios que foram parcialmente, ou totalmente, comprados pela fabricante (também chamadas subsidiárias). Neste caso, o jogo resultante é *first-party*.
  - ***Third-Party*** [85,45%]: Estúdios mais independentes são considerados *third-party developers*. No entanto, normalmente possuem contratos com *publishers* para desenvolver um jogo sob demanda ou até receber aporte financeiro para um jogo próprio. Um estúdio *third-party* pode ser comprado por uma *publisher*, tornando-se *in-house developer*, porém continua operando de maneira autônoma em suas práticas e cultura.

- ***Indie*** [52,73%]: Há também os *indie game developers* que, embora não tenha uma linha clara na divisão deste grupo com os outros, ressalta-se alguns aspectos

que os diferenciam. Por definição, desenvolvimento *indie* de jogos é o negócio de fazer jogos digitais sem o suporte de uma *publisher* (PARKER, 2011) e com liberdade criativa (GRIL, 2008). Outras características são: times pequenos, baixo orçamento e distribuição online. Entretanto, desenvolvedores *indies* que não se encaixam nessas propriedades como o time por trás do jogo *Journey*, que obteve apoio financeiro e distribuição de uma *publisher*; *Bastion*, que foi publicado pela *Warner Bros Enterntaiment* e *The Witness* que teve um custo de desenvolvimento elevado.

- ***Funding* (financiamento)** [*external* 46,15% | *self* 54,72% | *crowdfunding* 13,46%]: As abordagens mais conhecidas de captação de fundos são oriundas de contratos com publicadoras ou estratégias de *crowdfunding*[6]. Em projetos *indies*, que necessitam menor aporte, os próprios desenvolvedores arcam com os custos, por vezes, contratando empresas de distribuição quando o jogo está finalizado.

- ***Publisher: External/Self* (publicadora)** [*external* 52,73% | *self* 53,70%]: O papel da *publisher* ou publicadora é variado, mas está principalmente relacionado a financiamento e supervisão. Grandes *publishers* também distribuem os jogos enquanto que as menores terceirizam este processo com distribuidoras ou até mesmo com outras *publishers* maiores. Outras atribuições estão relacionadas com licenças, localização dos jogos[7], manual do jogo, design da caixa, entre outras. Para uma *publishers*, há um alto risco envolvido ao financiar e distribuir um jogo, pois os *royalties* de cada cópia do jogo são pagos ao fabricante do hardware de uma só vez sem reembolso caso não haja sucesso nas vendas. O papel da *publisher* interfere diretamente no processo de desenvolvimento. Ao financiar um jogo, normalmente são estipulados *milestones* para acompanhar o progresso do produto. Além disso, algumas *publishers* possuem um departamento de controle de qualidade (*QA*) na qual o jogo é extensamente testado antes de ser publicado. Esse departamento também pode ser responsável pelo processo de certificação da plataforma fabricante.

6.1.4 Contexto em nível Técnico

Neste nível de contexto são descritos detalhes sobre aspectos técnicos do jogo:

- ***Intelectual Property* (propriedade intelectual)** [*port* 11,11% | *reboot* 3,70% | *sequence* 14,55% | *expansion* 1,85% | *mod* 1,85% | *new ip* 77,78%]: A propriedade intelectual também interfere no contexto do desenvolvimento de um jogo. Isto

[6] Financiamento coletivo em que pessoas interessadas no produto financiam previamente o projeto (WIKIPEDIA, 2016b).

[7] Preparação do jogo para ser distribuído em uma região com cultura diferente.

porque um projeto que envolva uma remasterização de um título antigo ou uma adaptação (*Porting*[8]) possuirá atividades diferentes do que criar um jogo sem uma base. Por exemplo, ao fazer uma adaptação, iterações em busca do elemento *fun* e game design serão mínimas, enquanto que atividades mais técnicas serão predominantes. Para este contexto foram definidas seis divisões: ***Port***, quando o jogo está pronto e será portado para outra plataforma; ***Remaster / Reboot***, quando um jogo, normalmente antigo, é refeito, melhorando seus detalhes gráficos e de jogabilidade, geralmente graças a evolução do hardware; ***Franchise Sequence***, quando o jogo segue uma franquia já estabelecida; ***Expansion***, quando algum conteúdo extra é adicionado ao jogo original; ***Mod***, quando a comunidade de jogadores modifica elementos do jogo original; ***New Ip***, quando o jogo é desenvolvido do zero.

- ***Game Engine*** [*in-house* 24,07% | *in-house ready* 25,45% | *off-the-shelf* 51,85%]: Com o constante avanço da tecnologia, os jogos se tornando cada vez mais realistas, formas de lidar com a complexidade crescente surgiram diante da demanda e as *engines* são frutos desse movimento (ANDERSON *et al.*, 2013). *Engines* são suítes de desenvolvimento que diminuem a barreira inicial para criação de jogos oferecendo ferramentas parametrizadas, permitindo ao desenvolvedor focar mais no *game design* do que em detalhes técnicos. Por outro lado, estas facilidades embaraçam a capacidade de customização, o que pode impedir a construção de uma mecânica diferenciada para o jogo. Neste contexto foram definidas três situações: ***In-House (new)***, quando a engine foi construída do zero pelo time; ***In-House Ready***, quando a *engine* já estava pronta antes do início do desenvolvimento; e ***Off-The-Shelf***, quando o time usa uma solução de terceiros, já pronta.

6.1.5 Contexto em nível de Plataforma

Neste nível de contexto são descritos detalhes referentes a plataforma ou hardware em que o jogo será projetado. O jogo ainda pode ser multi-plataforma, abrangendo várias, senão todas as alternativas.

Alguns fabricantes de hardware possuem controles rígidos de que o jogo deve obedecer para ser oferecido naquela plataforma específica. Para estes detalhes é dado o nome de “certificação”, na qual cada fabricante possui um *checklist* diferente. Por exemplo, para a desenvolvedora ter seu jogo funcionando no console e, consequentemente, oferecido na loja virtual da *Sony*, os itens do *checklist* da certificação precisam estar completamente satisfeitos.

[8] Porting é adaptar um programa/jogo para uma plataforma diferente da qual o mesmo foi concebido.

É notório, então, que a plataforma alvo do jogo adiciona etapas no desenvolvimento. Ainda, quanto maior o número de plataformas, maior o esforço do time para obedecer as diferentes regras de cada fabricante. *Sony*, *Microsoft* e *Nintendo* possuem um processo de certificação mais complexo, enquanto que plataformas como *Windows* e *Linux* geralmente são mais acessíveis aos desenvolvedores.

As plataformas definidas no contexto são Consoles: **Microsoft** [29,90%], **Sony** [40%] e **Nintendo** [11,11%]; PC: **Windows** [65,45%], **MacOS** [40,00%] e **Linux** [29,63%]; **Mobile**: **Android** [25,45%] e **iOS** [38,18%].

6.1.6 Contexto em nível de *Game Design*

Este nível de contexto se refere a detalhes do tipo do jogo, ou seja, ao produto e seus elementos relacionados ao *game design*. O gênero (**Genre**) do jogo diz muito sobre como ele foi desenvolvido. Por exemplo, criar um jogo de esportes envolve muito trabalho com detalhes técnicos como a física, pois a ação deve se parecer com o mundo real. Enquanto que um *adventure* ou *rpg* possuem forte apelo em narrativa. O mesmo vale para o modo (**Mode**) do jogo que, sendo *singleplayer*, não vai exigir tanto esforço técnico comparado a criação de um ambiente *multiplayer*, sendo ele online ou local.

Os atributos definidos para o contexto são: Gênero: **Action** [27,78%], **Action-advendure** [25,93%], **Adventure** [5,56%], **Role-playing** [16,36%], **Simulation** [7,41%], **Strategy** [12,96%], **Puzzle** [20,37%] e **Sports** [3,70%]; Modo: **Single-player** [96,36%], **Multi-player** [18,52%] e **Multiplayer-online** [24,07%].

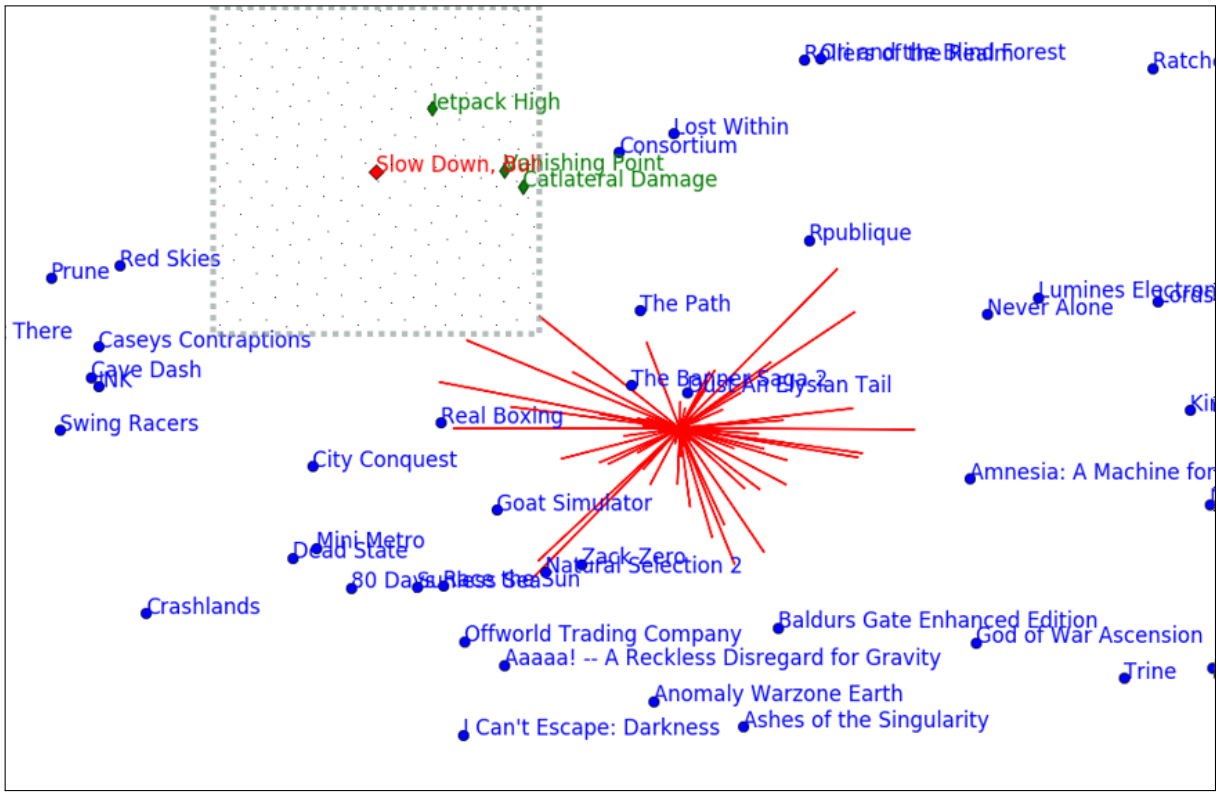
6.2 ALGORITMO DE SIMILARIDADES

A tabela de contexto, devidamente completa, serve de entrada para uma nova etapa do processo de recomendação: o *script* de similaridades. Este algoritmo utiliza uma técnica de reconhecimento de padrões ***Principal Component Analysis*** ou simplesmente PCA, utilizada para visualizar e quantificar relações entre muitas variáveis. Com esta abordagem é possível utilizar o conjunto de variáveis da tabela de contexto e visualizá-las em um plano 2D e, por fim, **verificar o grau de similaridade entre as amostras**.

Para a criação desta funcionalidade, modificou-se um *script*, escrito na linguagem *Python*[9], que permite visualizar o grau de correlação entre as amostras. De forma simplória, a execução do algoritmo consiste em receber a tabela de contexto como entrada (ver Apêndice D), aplicar o método *PCA* e criar uma visualização do mesmo utilizando o gráfico *bi-plot*.

[9] O script base está diponível no *github* (*https://github.com/teddyroland/python-biplot*) enquanto que o algoritmo modificado está disponível no endereço: *https://polako.github.io/dissertation/scripts*.

Figura 9 – Gráfico *PCA* com o contexto do jogo *"Slow down, Bull"* como entrada.



Fonte: elaborado pelo autor.

Para tornar o algoritmo mais completo, foi adicionada uma função que **recebe como entrada os dados sobre o perfil da equipe de desenvolvimento**, definido pelo desenvolvedor, com os mesmos atributos utilizados na tabela de contexto, e o gráfico resultante salienta os projetos que mais se assemelham ao contexto de entrada. Além do gráfico, o *script* retorna de sua execução a lista dos projetos mais similares.

A Figura 9 auxilia o entendimento do *biplot PCA*. Neste caso, o contexto de entrada foi o jogo *Slow down, Bull* (contexto na Listagem 2). Na figura, cada ponto do gráfico é uma amostra (jogo). Quanto mais próximos, mais semelhantes são suas características. O jogo de entrada, *Slow down, Bull*, está em vermelho, enquanto que os mais próximos estão em verde: *Catlateral Damage, Jetpack High, Vanishing Point*. O grau de proximidade é definido por uma constante, que nesta execução foi definida como "0.5".

6.3 JUNÇÃO DOS PROCESSOS SIMILARES

Com o *script* de similaridades retornando uma lista de projetos semelhantes, é possível realizar a **concatenação das atividades e práticas de cada um, formando um novo processo**. Porém, apenas colocar todos os elementos do processo juntos e gerar um fluxo não é muito relevante para uma equipe de desenvolvimento que vai

Listagem 1 – Segmento do modelo de abstração ilustrando o termo *testing*.

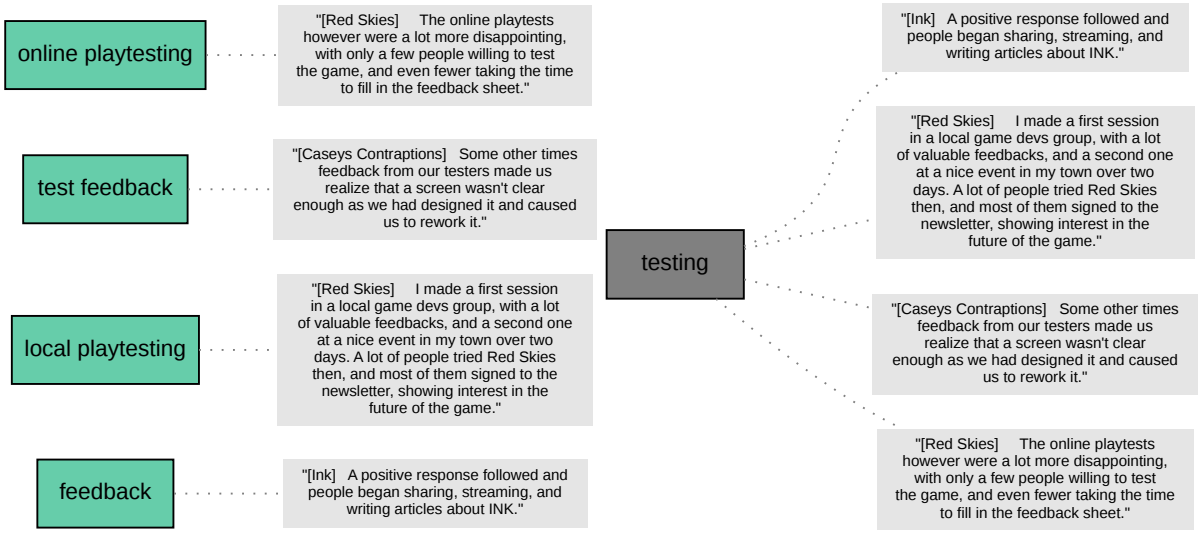
```
1  * testing
2    * local playtesting
3    * **online playtesting**
4    * usability testing
5    * **QA**
6    * frequent playtesting
7    * early playtesting
8    * **final testing**
9    * device testing
10   * early playtesting
11   * **final playtesting**
12   * **small QA**
13   * **playtesting**
14   * playtesting
15   * extra QA
16   * usability testing
17   * in-house QA
18   * **usability lab**
19   * **qa**
20   * alpha testing
21   * beta testing
```

iniciar um novo projeto. A ferramenta de recomendação deve mostrar o que aconteceu em outros projetos mas de forma sucinta. Não só isso, a ferramenta também deve levar em consideração o perfil do time, ou seja, **receber as informações de como o time pretende proceder durante o desenvolvimento e juntar com as informações dos projetos similares, resultando em um processo novo, que leva em consideração o que aconteceu no passado e as expectativas para o futuro**.

Há, entretanto, um problema, pois a estrutura de "chave:valor", definida na Seção 5.3, não definiu chaves equivalentes para as tarefas e características de mesma finalidade. Isso acarreta que, algumas vezes, os índices que definem elementos iguais podem possuir nomes diferentes. Para resolver esta questão, todos as 55 notas dos *postmortems* foram analisadas novamente e, para cada item, foi definido uma chave única. Por exemplo, no trecho da Listagem 1, as notas extraídas dos *postmortems* que dizem respeito a *"testing"* (linha 1) estão listadas nas linhas 2 a 21. Dessa forma, foi criado um grande dicionário que agrupa os nomes dos elementos para índices mais abstratos, chamado de **modelo de abstração**.

A primeira função do algoritmo[10], criado para junção dos processos, é pegar este modelo de abstração como entrada e transformá-lo em um dicionário em memória. A próxima etapa na execução é receber a lista de projetos similares, conforme o resultado da execução do *PCA*. Com essa lista, percorre-se os arquivos de notas e concatena-se, resultando em apenas um grande arquivo com as notas e citações de todos os projetos similares. Para melhor entendimento, foi adicionado o nome do jogo à citação, entre colchetes. Este arquivo é então analisado (*parser*) e cada elemento vai ser comparado e misturado com o modelo de abstração, gerando um novo nome para o elemento mas permanecendo com a mesma citação. Este exemplo pode ser melhor entendido na

Figura 10 – Elemento abstrato *testing* e as citações dos elementos agrupados.



(a) Elementos sem o dicionário de abstração. (b) Elementos com o dicionário de abstração.

Fonte: elaborado pelo autor.

Figura 10, que ilustra o motivo da criação do modelo de abstração. A Figura 10a mostra como ficariam os elementos no processo gerado sem o uso de um modelo que abstrai o nome dos elementos. A Figura 10b demonstra o uso do dicionário de abstração, que reduziu os quatro tipos de atividades referentes a teste para somente uma descrição, mantendo as citações.

6.3.1 Características do time de desenvolvimento no Sistema de Recomendação

Apenas concatenar os elementos não agregaria ao desenvolvedor, o processo resultante deve ser customizado de acordo com os preceitos e características do time. Para isso, **criou-se um formulário de entrada para capturar os dados de contexto** e também saber qual estrutura o processo final teria (ver formulário completo no Apêndice C). Então, por meio de uma série de perguntas o desenvolvedor vai moldando o tipo de processo a ser utilizado no projeto. Por exemplo, a pergunta abaixo indaga ao desenvolvedor sobre a fase de pre-produção que, pode ser cortada ou não do processo resultante.

> *"O time prefere trabalhar em uma fase exploratória, criando protótipos e testando hipóteses na busca pelo fator "diversão"? Ou Prefere entrar direto em desenvolvimento (full development) sem validações de protótipos, deixando feedbacks para outra fase?"*

[10] O *script* completo pode ser visualizado no site do projeto no endereço: *https://polako.github.io/dissertation/scripts*.

A estrutura do processo final recebeu dois modelos (*templates*) distintos para a fase de produção. Um com um viés **adaptativo**, focado em entregas contínuas de *features* completas e outra mais tradicional, semelhante ao método ***waterfall***, chamado de **preditivo**, com foco em um maior planejamento, documentação e entregáveis completos, iterando somente em fases de correção de *bugs*. A ferramenta de recomendação propicia ao desenvolvedor gerar ambos modelos, que é escolhido conforme a resposta da seguinte pergunta, no formulário de entrada de contexto:

> *" O time pretende desenvolver o jogo de forma incremental, agregando funcionalidades completas no decorrer do desenvolvimento, abraçando mudanças e adições de novas funcionalidades? Ou planejar previamente todas (ou a maior parte) das features, documentar e seguir estritamente esse plano?"*

6.4 O PROCESSO RESULTANTE

Por fim, como resultado da execução de todas as etapas do sistema de recomendação, um exemplo do processo final pode ser visualizado nos seguintes conjuntos de imagens[11]: na Figura 11 é mostrada a fase de pré-produção e na Figura 12 a fase de produção. Como contexto de projeto foram utilizados os dados do jogo *"Slow down, Bull"*, descrito na Listagem 2.

Neste exemplo, devido ao contexto do projeto ser voltado para metodologias ágeis, o algoritmo utilizou o modelo para processos **adaptativos**. O processo gerado também não contém atividades na fase de pós-produção.

A Figura 11 apresenta a fase de pré-produção com as atividades: **"requisitos e restrições"** ligada a uma citação oriunda do jogo *Vanishing Point*; **"fase de exploração"** também com uma citação do jogo *Vanishing Point*; **"prototipagem"** dos jogos *Catlateral Damage* e *Jetpack High*; **"planejamento de documentação"** do jogo *Vanishing Point*; e, finalizando a fase, a atividade de **"planejamento de *milestones*"** também do jogo *Vanishing Point*.

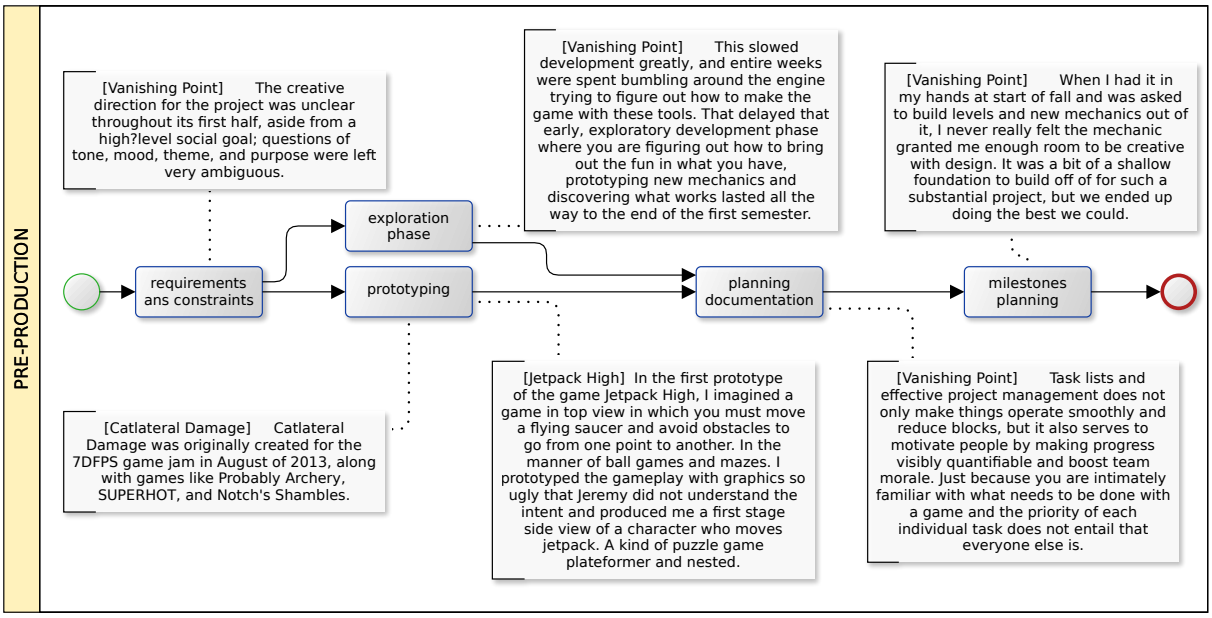
Na Figura 12 o fluxo do processo na fase de produção começa com a atividade de escolha de uma *feature* ou bug, dependendo do tipo de iteração, ou seja, se for para entregar um componente ou resolver um eventual problema. Esta atividade é fixa no modelo padrão **adaptativo**. Esta atividade recebe como entrada uma lista de propriedades do jogo que, neste caso, foram previamente pensadas na fase de pré-produção. Há, no entanto, a possibilidade de não haver a fase de pré-produção. Porém, a lista de *features/bugs* ainda permanece no processo pois é elemento vital para o

[11] Devido ao tamanho da imagem gerada, o processo foi redesenhado em notação *BPMN*.

Listagem 2 – Contexto do projeto *"Slow down, Bull"*.

```
* [Activities] Agile: yes
* [Activities] Prototyping: yes
* [Activities] Performance Optimization: no
* [Activities] Tools Development: no
* [Activities] Outsourcing: Assets: no
* [Activities] Outsourcing: Work: yes
* [Activities] Pre-Production: Short: no
* [Activities] Pre-Production: Long: yes
* [Activities] Post-Production: Normal: yes
* [Activities] Post-Production: Heavy: no
* [Activities] Reuse: Code: no
* [Activities] Reuse: Assets: no
* [Activities] Testing: In-Ouse Qa Team: yes
* [Activities] Testing: Closed Beta (user): no
* [Activities] Testing: Open Beta (user): no
* [Activities] Testing: Early Access (user): no
* [Activities] Marketing/Pr: Self: no
* [Activities] Marketing/Pr: Outsourced: no
* [Team] Size: <=5: yes
* [Team] Size: 5 - 25: no
* [Team] Size: >25: no
* [Team] Type: Single: yes
* [Team] Type: Collaborative: no
* [Team] Distribuited: yes
* [Project Management] Developer Type: First-Party: no
* [Project Management] Developer Type: Second-Party: no
* [Project Management] Developer Type: Third-Party: yes
* [Project Management] Indie: yes
* [Project Management] Funding: External: yes
* [Project Management] Funding: Self: no
* [Project Management] Funding: Crowdfunding: no
* [Project Management] Publisher: External: no
* [Project Management] Publisher: Self (same Developer): yes
* [Technical] Intelectual Property: Port: no
* [Technical] Intelectual Property: Remaster / Reboot: no
* [Technical] Intelectual Property: Franchise Sequence: no
* [Technical] Intelectual Property: Expansion: no
* [Technical] Intelectual Property: Mod: no
* [Technical] Intelectual Property: New Ip: yes
* [Technical] Engine: In-House (new): no
* [Technical] Engine: In-House Ready: no
* [Technical] Engine: Off-The-Shelf: yes
* [Platform] Console: Microsoft: no
* [Platform] Console: Sony: no
* [Platform] Console: Nintendo: no
* [Platform] Pc: Windows: yes
* [Platform] Pc: Mac: no
* [Platform] Pc: Linux: no
* [Platform] Mobile: Android: no
* [Platform] Mobile: Ios: no
* [Design] Genre: Action: no
* [Design] Genre: Action-Adventure: no
* [Design] Genre: Adventure: no
* [Design] Genre: Role-Playing: no
* [Design] Genre: Simulation: no
* [Design] Genre: Strategy: no
* [Design] Genre: Puzzle: yes
* [Design] Genre: Sports: no
* [Design] Mode: Single-Player: yes
* [Design] Mode: Multi-Player (offline): no
* [Design] Mode: Multi-Player Online: no
```

Figura 11 – Fase de pre-produção do processo recomendado pelo algoritmo para o contexto do jogo *Slow down, Bull*.



Fonte: elaborado pelo autor.

ciclo de iterações do desenvolvimento. Esta lista tem papel muito semelhante ao *sprint backlog* utilizado no método ágil *Scrum*.

O fluxo do processo continua com o **loop de iterações do desenvolvimento**. Este elemento marca o início das atividade de produção e também é elemento fixo no modelo de processo adaptativo padrão. Neste caso, duas citações retiradas dos *postmortems* dos jogos *Vanishing Point* e *Catlateral Damage* são destacadas. A principal diferença entre o modelo padrão iterativo e preditivo está neste elemento, que funciona de forma semelhante ao *sprint* dos métodos ágeis. A premissa é de que o processo seja desenvolvido em ciclos curtos de entregas contínuas na forma de componentes completos. Ou seja, a cada fim de iteração, um componente completo deve ser agregado ao jogo existente. O artefato **entrega de feature** representa este componente.

Dentro das iterações é que as atividades em nível técnico acontecem. No processo gerado para o contexto do jogo *Slow down, Bull*, há quatro atividades explicitamente descritas: **polimento e refinamento**, ligada a uma citação do jogo *Vanishing Point*; **reuniões**, ao jogo *Jetpack High*; **refatoração do desenvolvimento** também ao jogo *Jetpack High*; e **tarefas de design** ligada a duas citações do jogo *Catlateral Damage*.

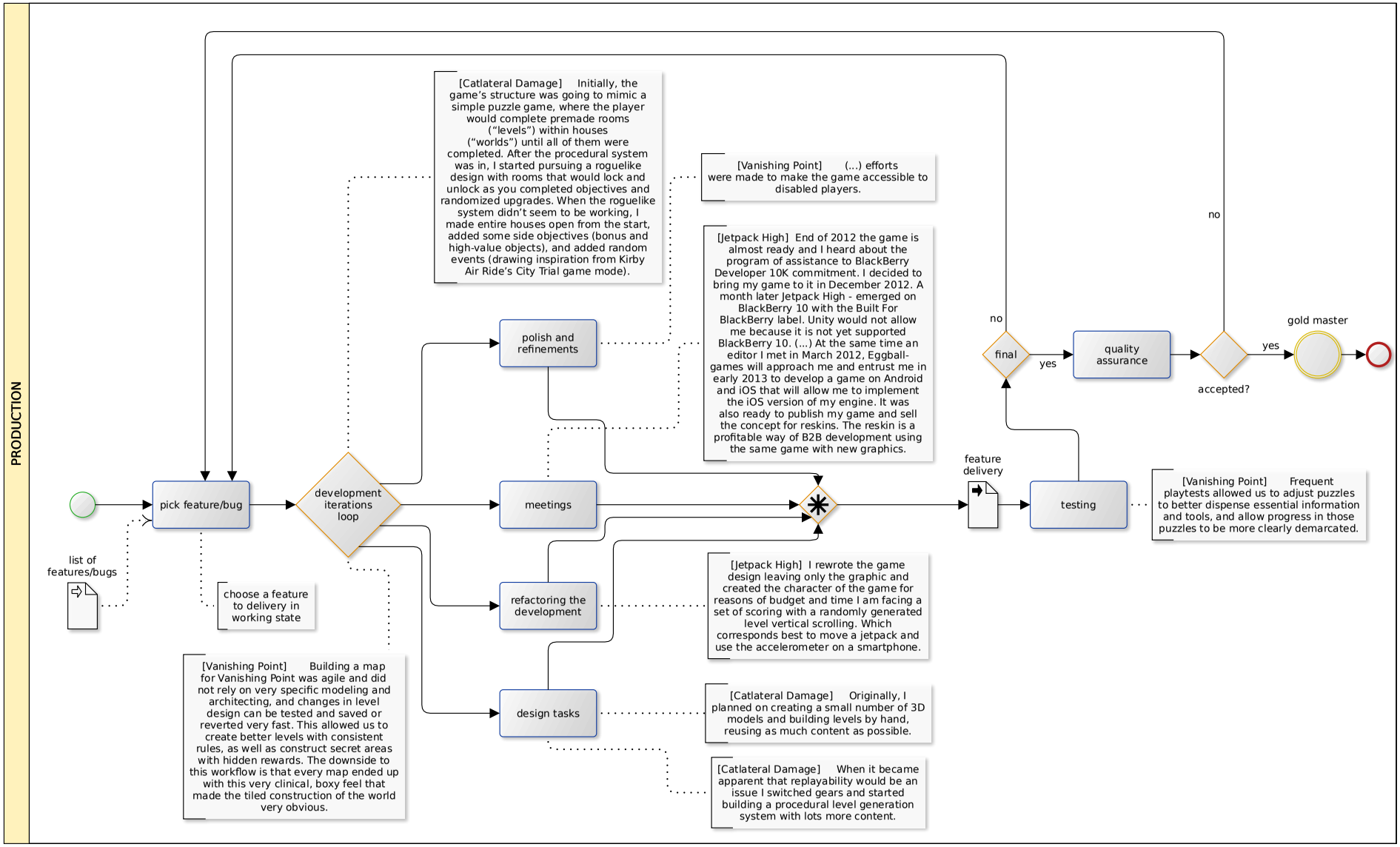
É importante destacar que não há a intenção de propiciar uma granulosidade fina na definição das tarefas, muito menos detalhar todas as atividades a serem realizadas pelo time de desenvolvimento. Detalhes neste nível de precisão dependem de uma

fonte mais acurada de informações sobre o desenvolvimento do que *postmortems*.

Seguindo o fluxo de desenvolvimento do processo gerado, após a entrega do componente da iteração, inicia-se a atividade de **teste**. Esta etapa é essencial para o processo do jogo. Outrossim, ela pode acontecer das mais variadas formas. Por exemplo, o time pode dispor de um laboratório de testes de jogabilidade com usuários, um time especializado em testes, testes automatizados, testes com usuários, um grupo fechado de testadores beta, entre vários outros. Neste caso, apenas a citação do jogo *Vanishing Point* foi agregado ao elemento. Esta atividade acontece dentro do *loop* de iterações, embora esteja fora. Isto foi feito propositalmente, pois o teste (ou certo tipo de teste) pode acontecer somente em certas versões do jogo (*builds*) e o processo recomendado não tem o intuito de restringir tal liberdade.

Após o fim das iterações e após o jogo estar devidamente testado, o fluxo determina um teste, na qual avalia se todas as iterações foram completadas. Em caso positivo, segue-se para a atividade de **garantia de qualidade**. Esta normalmente se confunde com testes e pode ser usada de forma intercambiável. No entanto, no modelo padrão é definido que este elemento detalha a revisão do jogo por uma entidade externa, seja ela uma publicadora ou uma plataforma (*Sony*, *Steam*, *Microsoft*). De forma geral, o que se verifica aqui são inconformidades junto aos padrões exigidos pelas mesmas. Caso a versão (*build*) final esteja de acordo as exigências, denomina-se **gold master** e publica-se o jogo.

Figura 12 – Fase de produção do processo recomendado pelo algoritmo para o contexto do jogo *Slow down, Bull*.



Fonte: elaborado pelo autor.

6.5 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou a parte principal referente a construção do sistema de recomendação. Primeiro foi definido um contexto que defina um projeto de jogos digitais. Este contexto é composto por um conjunto de 61 características que definem como o desenvolvimento do jogo pode acontecer. Após a definição, foram analisados os 55 projetos e extraídos os seus atributos para uma tabela de contexto.

A segunda parte da construção do sistema consistiu na criação do algoritmo que destaca os projetos mais similares, efetua a junção dos elementos destes processos, analisa se estavam de acordo com o que foi especificado no formulário de contexto preenchido pelo time de desenvolvimento e, por fim, fazia a junção dos elementos formando um novo processo, utilizando um dos dois tipos de modelos padrões definidos: adaptativo ou preditivo.

6.6 DISCUSSÃO DOS RESULTADOS DO CAPÍTULO

Quando se inicia um projeto de software há várias preocupações relacionadas aos requisitos e detalhes técnicos do time. Por exemplo, detalhes como o objetivo da aplicação, quando se refere ao cliente, ou o tipo de *framework* a ser utilizado, quando se fala em aspectos técnicos no time de desenvolvedores. Este detalhes são relativos e, portanto, variam conforme o contexto da aplicação. O mesmo vale para outras aplicações, como jogos digitais.

Durante a pesquisa, não foram encontrados materiais que se referissem a definição de um contexto para jogos digitais. Isto acarretou na necessidade da concepção de um contexto que definisse, de forma geral, os principais atributos no desenvolvimento de um projeto de um jogo digital.

Definiu-se então, 61 atributos para o contexto. Este conjunto de características não tem a intenção de ser imutável. Mas sim de servir como base para uma extensão ou eventual refatoração. Inclusive, a agregação de variáveis a este conjunto é encorajada, visto que não interfere no funcionamento do sistema de recomendação.

O sistema de recomendação é a principal contribuição deste trabalho. O intuito do processo gerado é auxiliar o time de desenvolvimento na fase de planejamento da produção do jogo, ou seja, antes mesmo de começar a pré-produção. O processo resultante vai ajudar o desenvolvedor a tomar decisões de planejamento do processo ou até, em alguns casos, servir como um processo em alto nível a ser seguido.

Por exemplo, antes de começar o desenvolvimento de um jogo, preenche-se o formulário de contexto com o jogo a ser produzido. Em seguida executa-se o sistema de recomendação, gerando um novo processo recomendado tendo como base os projetos

de jogos com contextos similares e características do time e do projeto, definida no contexto. O engenheiro de software ou responsável pela equipe e jogo pode analisar os equívocos e acertos ocorridos em outros projetos de mesmo contexto. Com estes dados, é possível traçar planos que previnam tais acontecimentos ou a repetição de práticas que deram certo.

7 VALIDAÇÃO

Este Capítulo apresenta as etapas de validação do sistema de recomendação, descrito no capítulo anterior. São duas etapas complementares: a Seção 7.1 descreve a escolha dos projetos analisados e a validação dos mesmos conforme o grau de semelhança entre o processo extraído o processo gerado (recomendado) pelo sistema. A Seção 7.2 detalha a validação dos processos recomendados por parte dos autores dos *postmortems*[1]. Por fim, a Seção 7.3 resume os principais pontos do capítulo.

7.1 VALIDAÇÃO “A”: SEMELHANÇA DO PROCESSO GERADO

Para realizar esta etapa, descrita no capítulo 4, quatro diferentes projetos foram escolhidos, com um grau de similaridade baixo (pontos distantes no gráfico *biplot PCA*). Os projetos são dos seguintes jogos: *Prune*, *Offworld Trading Company* , *NFL Rush Heroes & rivals* e *Slow Down, Bull*.

Conforme pode ser verificado na Figura 13, foram realizados quatro *Biplots PCA* utilizando como entradas os projetos anteriormente citados. Cada execução resultou em um gráfico mostrando a lista de projetos semelhantes. O resultado do projeto *Prune* está na Figura 13a; do projeto *Offworld Trading Company* está na Figura 13b; do projeto *NFL Rush Heroes & rivals* está na Figura 13c e do projeto *Slow Down, Bull* na Figura 13d.

Na próxima etapa da validação a ferramenta de recomendação foi executada, gerando o processo recomendado para cada uma das entradas (projetos).

Para fazer uma comparação entre os elementos gerados pelo algoritmo, modificou-se o *script* para que também gerasse uma tabela com os elementos do processo. Conforme o exemplo na Tabela 4, os elementos do processo foram descritos, juntamente com as respectivas citações e fases. Com o auxílio da Tabela 2, cada um dos elementos de todos os quatro processos recomendados foram analisados e o resultado anotado na última coluna (“Resultados”).

Os resultados mostram uma precisão de mais de 73% para o processo recomendado para o contexto do jogo *Offworld Trading Company* como o melhor resultado, sendo que o projeto do jogo *NFL Rush* teve apenas 53% de precisão. O *recall* mais baixo foi de 80% enquanto que os demais não baixaram de 93%.

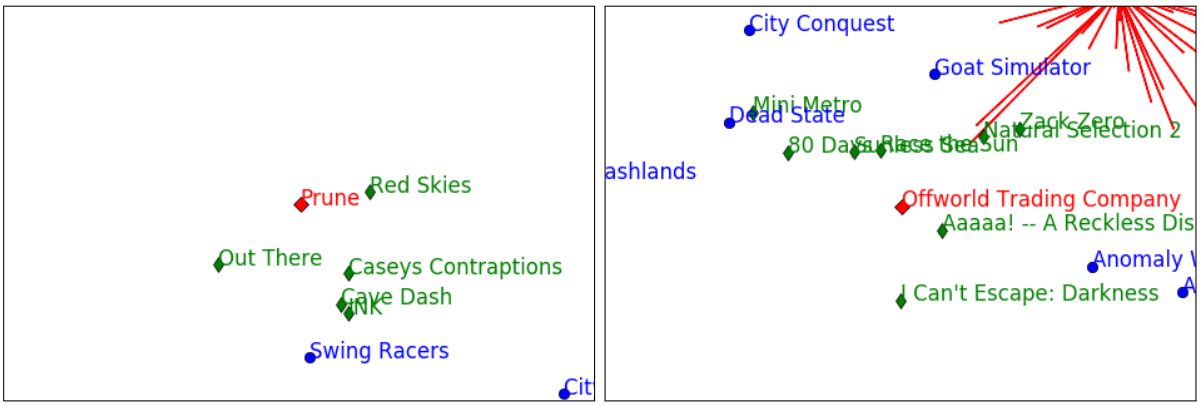
Embora os processos gerados tenham uma quantidade elevada de elementos “falsos positivos”, a maioria dos elementos contidos no processo gerado foram “verda-

[1] Todos os dados da validação estão no site do projeto no endereço: *https://polako.github.io/dissertation/validation*.

Figura 13 – *Biplots PCAs* dos projetos analisados demonstrando a similaridade.

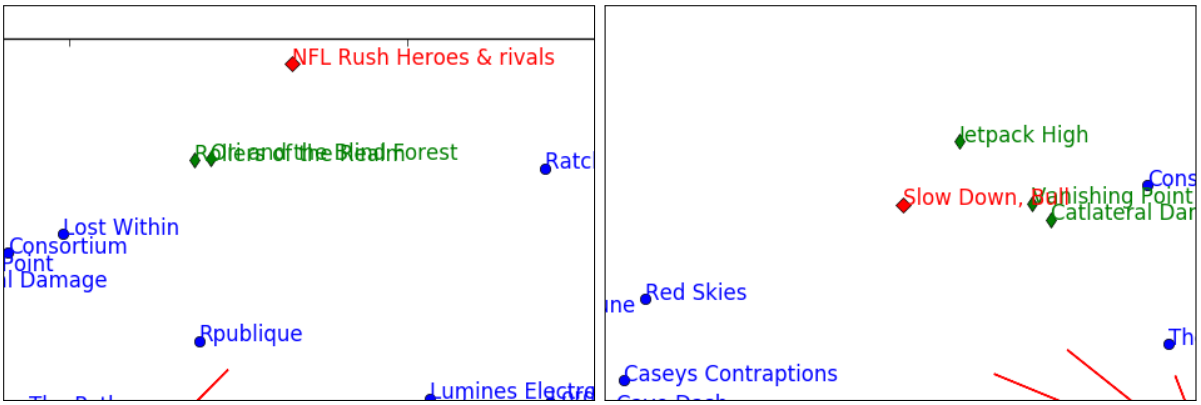
(a) *Biplot* PCA do contexto do jogo ***Prune***. Projetos similares: *Red Skies, Out There, INK, Cave Dash, Caseys Contraptions*.

(b) *Biplot* PCA do contexto do jogo ***Offworld Trading Company***. Projetos similares: *Mini Metro, Sunless Sea, Race the Sun, I Cant Escape: Darkness, 80 Days, Natural Selection 2, Aaaaa! – A Reckless Disregard for Gravity, Zack Zero*.



(c) *Biplot* PCA do contexto do jogo ***NFL Rush Heroes & rivals***. Jogos similares: *Ori and the Blind Forest, Rollers of the Realm*.

(d) *Biplot* PCA do contexto do jogo ***Slow Down, Bull***. Jogos similares: *Vanishing Point , Catlateral Damage , Jetpack High*.



Fonte: elaborado pelo autor.

deiros positivos". Isto mostra que o sistema de recomendação acertou, mais do que errou, na concatenação dos elementos dos projetos similares.

7.1.1 Análise do fluxo dos elementos do processo

Esta subseção detalha a última parte da primeira etapa de validação que consiste na análise do fluxo de elementos da processo. Para cada jogo analisou-se as disparidades, referentes ao fluxo das atividades, entre o processo extraído e validado do *postmortem* e o processo gerado pelo sistema de recomendação.

Tabela 4 – Tabela de validação, *precision-recall*, para o projeto *"Slow Down, Bull"*. Falsos negativos: *task managements, tasks review, log documentation, engine unity*.

Fase	Elemento	Citação	Resultado
characteristics	scope	[Vanishing Point] Reducing the . . .	TP
characteristics	development problems	[Vanishing Point] Crunching under a . . .	FP
characteristics	budget details	[Catlateral Damage] Overall, our . . .	FP
characteristics	project duration	[Catlateral Damage] (...) project du . . .	FP
characteristics	development problems	[Catlateral Damage] And since we . . .	TP
characteristics	budget details	[Jetpack High] To finance the pro . . .	TP
characteristics	scope	[Catlateral Damage] Like any pro . . .	TP
pre-production	requirements and constraints	[Vanishing Point] The creative direc . . .	FP
pre-production	exploration phase	[Vanishing Point] This slowed deve . . .	TP
pre-production	milestones planning	[Vanishing Point] When I had it in . . .	FP
pre-production	planning documentation	[Vanishing Point] Task lists and ef . . .	TP
pre-production	prototyping	[Catlateral Damage] Catlateral Da . . .	TP
pre-production	prototyping	[Jetpack High] In the first prototype . . .	TP
production	development iterations loop	[Vanishing Point] Building a map for . . .	TP
production	testing	[Vanishing Point] Frequent playtests . . .	TP
production	polish and refinements	[Vanishing Point] (...) efforts were . . .	TP
production	design tasks	[Catlateral Damage] Originally, I . . .	FP
production	design tasks	[Catlateral Damage] When it be . . .	FP
production	development iterations loop	[Catlateral Damage] Initially, the . . .	FP
production	refactoring the development	[Jetpack High] I rewrote the game . . .	TP
production	meetings	[Jetpack High] End of 2012 the . . .	FP
team	general team details	[Vanishing Point] Furthermore, . . .	FP
team	general team details	[Catlateral Damage] We worked . . .	TP
team	outsourcing	[Jetpack High] Looking for a graphic . . .	TP
team	collaborative	[Catlateral Damage] (...) started wor . . .	TP
team	general team details	[Catlateral Damage] Towards the . . .	TP
team	outsourcing	[Vanishing Point] One of the things . . .	TP

Fonte: elaborado pelo autor.

Tabela 5 – Resultado total do cálculo *Precision* e *Recall* para os elementos dos quatro processos gerados.

	Slow Down, Bull	NFL Rush	Offworld Trad. C.	Prune
Precision	62,96%	53,66%	73,68%	56,34%
Recall	80,95%	95,65%	96,55%	93,02%

Fonte: elaborado pelo autor.

7.1.1.1 Prune

- **Processo extraído:** O processo do projeto Prune tem como elementos mais notórios a pre-produção longa, extenso uso de prototipagem, e iterações de planejamento, prototipagem e testes antes de entrar em produção. Destaque também para o planejamento das tarefas mesmo que para um time de apenas um desenvolvedor.

- **Processo recomendado:** O processo gerado apresenta uma vasta fase de pre-produção. Neste caso o processo acerta, dando ênfase a requisitos e planeja-

mento. No entanto peca em salientar a prototipagem. Na fase de produção há algumas citações que dão a entender que há uma entrega contínua, o que não é verdade, como explicado anteriormente. Outros detalhes que não condizem com o que foi dito pelo desenvolvedor no *postmortem* são detalhes referentes a *engine* e ferramentas e desenvolvimento paralelo. Por fim, aspectos relacionados ao time estão muito parecidos com a realidade enfrentada pelo desenvolvedor do jogo *Prune*. Ou seja, muita ênfase em *outsourcing* e destaque para o tamanho pequeno do time.

7.1.1.2 Offworld Trading Company

- **Processo extraído:** o processo de offworld se destaca pela ausência da fase de pre-produção. O time decidiu ir direto para o desenvolvimento entregando builds continuamente para a comunidade, recebendo *feedbacks* e adicionado mais features a cada iteração.

- **Processo recomendado:** o processo gerado para offworld é adaptativo, sem a fase de produção, com enfase nas atividades relacionadas ao marketing e teste, o que reflete o processo extraído originalmente. Destaque também para o *vertical slice*, que podem ser traduzidos como versões a serem submetidas a testes com usuários, concordando com o que foi extraído do *postmortem*.

7.1.1.3 NFL Rush Heroes & rivals

- **Processo extraído:** o processo de NFL foi o que recebeu maior número de modificações no *feedback* do desenvolvedor, inclusive o próprio re-desenhou parte do processo para torná-lo mais fiel. Consiste em uma grande fase de pré-produção focada em pesquisa, prototipagem e a busca pela diversão. A etapa de produção só inicia depois da extensa fase de experimentações, com as tarefas e prazos já estabelecidos.

- **Processo recomendado:** no novo processo, na pre-produção, os elementos estão conceitualmente corretos, como planejamentos de marcos. No entanto, algumas citações fazem referência a atividades que não aconteceram. O mesmo vale para a fase de produção. Ainda nesta fase, a atividade de refinamento do jogo não consta no *postmortem* escrito pelo autor.

7.1.1.4 Slow Down, Bull

- **Processo extraído:** na fase de pre-produção do processo original extraído do *postmortem* consiste basicamente de iterações entre prototipagem e *playtesting*

em busca de do elemento diversão. Na fase de produção também há iterações com experimentos com ênfase em documentação e atividades de gerenciamento.

- **Processo recomendado:** no novo processo gerado, a fase de pre-produção possui a atividade de exploração, refletindo exatamente o que foi relatado pela autora. No entanto, as atividades de documentação não condizem com o que foi extraído do *postmortem*. Na etapa de produção, como o processo gerado é adaptativo, reflete bem o que a autora descreveu como sendo seu processo. As exceções são as atividades de refinamento e refatoração, que não são citadas no *postmortem*.

7.2 VALIDAÇÃO “B”: VIABILIDADE DOS PROCESSOS

A segunda etapa da validação é complementar a primeira, na qual, juntamente com os autores dos projetos analisados, foi solicitado para darem seus pareceres sobre o processo gerado pela ferramenta de recomendação, utilizando um formulário. O formulário pode ser visualizado, em sua completude, no Apêndice B.

7.2.1 Prune

O autor do *postmortem* e desenvolvedor do jogo *“Prune”* fez algumas ressalvas quanto a similaridade do processo gerado comparado ao que aconteceu na realidade. Como o processo gerado teve algumas atividades de documentação e planejamento o autor alertou que isto não ocorreu, visto que trabalhava sozinho (item h-1). O autor ainda rechaça o elemento de *vertical slice* como um marco durante o desenvolvimento (item h-2).

Um detalhe importante apontado pelo autor define o que determina se o jogo está terminado ou não. No processo gerado o jogo termina após uma série de correções de *bugs*, sem a adição de novas funcionalidades durante este período. Segundo o resposta, para o autor o que determina se o jogo está pronto é a sua qualidade, ou a sua completude em relação ao conceito inicial (item h-3).

(item h-1) *“In the pre-production stage, it is correct that I started with a concept and figured out some requirements and constraints and started prototyping. However, I didn’t really spend much or any time on milestone planning and my planning docs were very minimal since I worked on the game solo.”*

(item h-2) *“As far as production, I didn’t specifically aim to have a vertical slice. I suppose you could say I maybe effectively had a vertical slice about 6-7 months in since I had a small demo of the game that showed the core gameplay. But it wasn’t necessarily a development goal.”*

(item h-3) *“And then the section of the flow diagram where it loops back to the iteration stage based on if there ARE or ARE NOT bugs isn’t accurate. The determination of whether you’re finished isn’t just based on bugs but based on the quality of the current version of the game and how close it is to what you have in your head as far as the final product. Not sure how to rename this portion of the flow diagram... I guess the way I think of it internally is "Is it good? Am I happy with it?".”*

7.2.2 Offworld Trading Company

O autor do *postmortem* e desenvolvedor do jogo *“Offworld Trading Company”* foi sucinto ao concordar que o processo recomendado pelo sistema é similar ao que foi utilizado durante o desenvolvimento do jogo (item i-1).

No entanto, fez uma ressalva quanto o processo de *quality assurance*. Segundo ele, houve uma parceria desde o princípio do projeto e não somente após o jogo estar completo (item i-2). Neste caso pode ter ocorrido um equívoco, pois o termo *QA* as vezes é utilizado para designar times de testes, o que não é o caso aqui, onde a garantia de qualidade define o teste de conformidades do jogo por terceiros, como publicadoras ou plataformas.

(item i-1) *“Yes, this workflow looks similar to what we did with Offworld, in that we did a lot of iteration with our feature development.”*

(item i-2) *“One thing that is different is that QA was involved with feature development since we had them test all of our (roughly) 10 major releases during the Early Access project. We didn’t hold off from giving them the game until we were at our final release candidate.”*

7.2.3 NFL Rush Heroes & rivals

O autor do *postmortem* e desenvolvedor do jogo *“NFL Rush Heroes & rivals”* também foi sucinto ao dizer que o processo gerado é bem semelhante ao processo utilizado por ele e seu time durante do desenvolvimento do jogo (item j-1). Quando perguntado se usaria este processo em um jogo similar ao seu (*NFL Rush Heroes & rivals*), o mesmo foi enfático ao dizer que usaria algo “quase igual” para o desenvolvimento (item j-2).

(item j-1) *“Yes this is pretty similar to what we would use.”*

(item j-2) *“I’d use something almost exactly like this. I think you’ve nailed it. Good job!”*

7.2.4 Slow Down, Bull

A autora do *postmortem* e desenvolvedora do jogo *“Slow down, Bull”* respondeu da seguinte forma as quatro questões: Quanto a similaridade do processo gerado em relação ao que ocorreu durante o desenvolvimento a autora argumentou que os elementos estão muito abstratos para se ter uma noção do que realmente aconteceu (item k-1). Sobre o elementos e o sentido dos mesmos, a autora salienta que fazem sentido, mas de um “ponto de vista lógico” (item k-2). Segundo a autora, o fluxo do processo gerado não funciona bem para determinados tipos de eventos imprevistos bem como atividades que acontecem em paralelo (item k-3). Sobre a viabilidade em fazer uso do processo gerado, a autora acredita que a sua maior utilidade seria no planejamento prévio do projeto, mas não para formatar qualquer tipo de *pipeline* de produção (item k-4).

(item k-1) *“Sort of? But at a generalized enough to a point where it’s not significant.”*

(item k-2) *“Everything seems to make sense from a logical standpoint.”*

(item k-3) *“I don’t think the workflow accounts well for processes that form as a reaction to an unforeseen event, or processes that happen in parallel (like prototyping all throughout development)”*

(item k-4) *“(...) this might be a useful thing to look at in the beginning, but I would not use it to create a production pipeline because the circumstances have almost assuredly changed since.”*

(item k-5) *“I think the tool that you describe has some really interesting implications from an analytical standpoint, but it is not something I would use in a practical sense, like I wouldn’t use it to develop processes or pipelines from. In practice, every game requires different processes to make, and in fact the process used to make a game is as much a part of its design as the actual game. (...) The reasons for variation in process are vast and constantly changing. Even in big studios you tend to end up with processes that are perfect for making the game you were working on the year before (...) However, I still think it is valid work from an analytical standpoint, like as something to consider when having the conversation about dev process for the next game.”*

A autora continuou seu *feedback* ressaltando que o processo gerado é muito interessante de um ponto de vista analítico, mas não algo a ser usado na prática, ou seja, para construir um fluxo de atividades de produção. Para ela, cada projeto exige um processo diferente e o design do jogo implica em como esse processo será (item

k-5). Este ponto de vista vai de encontro com uma das hipóteses deste trabalho, de que o contexto do jogo (destacado na Seção 6.1), e aspectos referentes ao design, influenciam o comportamento do time de desenvolvimento e assim o processo a ser seguido.

7.3 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou as duas fases de validação do sistema de recomendação. Para tanto, foi utilizado quatro diferentes contextos, extraídos de quatro projetos de jogos, para que a ferramenta recomendasse um novo processo e então fosse possível realizar uma comparação do processo extraído e do processo recomendado.

A primeira etapa da validação foi o cálculo do *precision* e *recall* entre os elementos dos processos extraídos e dos processos recomendados.

A segunda etapa da validação foi realizada com o auxílio dos próprios desenvolvedores. Utilizando um formulário, os autores dos *postmortems* foram perguntados sobre a precisão do processo gerado quando comparado ao que os mesmos viveram durante o desenvolvimento do respectivo jogo. Também foi indagado sobre a viabilidade em usar o processo recomendado em um novo projeto, de mesmo contexto. De forma geral, as respostas foram positivas, concordando que o processo reflete o que foi feito e que poderia ser útil para auxiliar na execução de um novo projeto de jogo.

8 CONCLUSÕES

Este capítulo resume a pesquisa realizada neste trabalho, suas contribuições, limitações e trabalhos futuros. A Seção 8.1 resume os objetivos do trabalho e seus respectivos resultados. A Seção 8.2 resume as contribuições do trabalho. A Seção 8.3 explica algumas limitações do trabalho. Por fim, o capítulo termina com as perspectivas de trabalhos futuros na Seção 8.4.

8.1 REVISÃO DOS OBJETIVOS E RESULTADOS

Neste trabalho foi realizada a concepção de um sistema de recomendação de processos para desenvolvimento de jogos digitais. Este sistema foi criado com duas premissas principais. A primeira delas baseia-se no **aprendizado por meio de experiências prévias** de outras equipes de desenvolvedores. Essas experiências foram extraídas de uma base de dados de *postmortems*, escritas pelos próprios integrantes dos times de desenvolvimento, ressaltando detalhes do projeto do jogo digital. A segunda premissa é a **customização do processo** recomendado. O sistema leva em consideração as características do time de desenvolvimento para entregar um processo mais próximo à realidade e ao perfil dos desenvolvedores.

Para a construção do sistema de recomendação de processos para jogos digitais, alguns objetivos específicos foram definidos. Primeiramente, como descrito no Capítulo 5, houve a necessidade da criação de uma base de dados com os relatos de experiências dos desenvolvedores de jogos. Estes dados foram extraídos de artigos, escritos pela própria equipe de desenvolvimento, na forma de *postmortems*. Como estes artigos não possuem uma estrutura rígida, foi necessária a análise individual de mais de 100 artigos, com o intuito de extrair somente informações relevantes ao processo de desenvolvimento. Após a análise, foi realizada a extração, de forma automatizada, das informações textuais para então serem transformadas em diagramas gráficos, contendo as atividades e características do projeto bem como o fluxo de execução das mesmas. A base de dados totalizou **55 processos**, sendo que sete destes foram validados juntamente como os autores dos *postmortems*.

Para a construção do sistema de recomendação, primeiro definiu-se um **contexto para projetos de jogos digitais** (Seção 6.1). Uma lista de 61 características booleanas, divididas em seis grupos, que representam o tipo de projeto de jogo. Com estas variáveis, analisou-se novamente os projetos e foram extraídas as informações de cada um.

Utilizando estas informações, o **sistema de recomendação encontra os proje-**

tos mais similares utilizando uma técnica de reconhecimento de padrões, denominada *PCA* (Seção 6.2, 6.3 e 6.4). O sistema então gera um novo processo recomendado utilizando como base: a lista de projetos mais similares ao contexto de entrada e o perfil da equipe. Estas informações de entrada são coletadas com um formulário de contexto preenchido pela equipe de desenvolvimento.

Os processos gerados foram validados por quatro desenvolvedores (Capítulo 7). Dentre os quais, todos acharam a proposta válida, ou seja, o processo gerado pode auxiliar o time de desenvolvimento durante o projeto.

Durante a validação, a maioria dos desenvolvedores declarou que não há como repetir um processo de desenvolvimento em um jogo. A forma com que o mesmo vai ser desenvolvido vai depender das características do jogo. Nem mesmo a remasterização de um jogo vai utilizar o mesmo processo. Nas palavras da desenvolvedora do jogo *Slow down, Bull*, *"cada jogo possui um processo diferente"*.

Sendo assim, gerar um processo que atenda perfeitamente as atividades a serem realizadas é utópico. Portanto, o objetivo deste trabalho é auxiliar na concepção e planejamento de um processo, antes mesmo da fase de pré-produção. Deste modo, o processo gerado serve como subsídio para a tomada de decisões, após definido o contexto do jogo a ser produzido.

8.2 CONTRIBUIÇÕES DESTE TRABALHO

A síntese das principais contribuições deste trabalho são as seguintes:

- Criação de uma base de dados contendo informações sobre o processo de desenvolvimento de projetos de jogos digitais;
- Definição de um contexto de projeto cujo objetivo é definir os atributos do processo de desenvolvimento para jogos digitais, tais como perfil da equipe, detalhes técnicos, detalhes do processo de engenharia de software, questões de gerenciamento e design.
- Concepção de um sistema de recomendação de processos customizados para projetos de jogos digitais utilizando como base as experiências prévias de outros projetos de contexto semelhante e também o perfil da equipe de desenvolvedores.

8.3 LIMITAÇÕES DESTE TRABALHO

Neste trabalho houve uma preocupação com a validade dos resultados, principalmente em relação ao processos gerados, dado que o propósito dos *postmortems* não é descrever os elementos do processo. Este foi um dos motivos de haver duas

etapas de validação, uma na primeira extração do processo (Seção 5.5) e outra na validação dos processos gerados pelo sistema de recomendação (Seção 7).

Ainda sobre *postmortems*, nada garante que os dados contidos no texto demonstrem todas as atividades realizadas pelos desenvolvedores. Principalmente em empresas grandes, nas quais certas informações são intencionalmente omitidas.

Assim como exposto pela autora do jogo *Slow down, Bull*, em seu texto de *feedback* da validação (Seção 7.2.4), o processo gerado não consegue definir tarefas específicas a serem seguidas pelos desenvolvedores. Isto se dá pois as informações contidas na base de dados (*postmortems*) não tem tal especificidade. Como justificativa ao uso de *postmortems* como recurso, não foi encontrada outra fonte com informações mais precisas, disponível para pesquisa. Não obstante, o processo gerado não define tarefas específicas para o desenvolvimento do jogo, mas sim um conjunto de experiências, erros e acertos, vividas por outros desenvolvedores em projetos de jogos de contexto semelhante.

8.4 PERSPECTIVAS PARA TRABALHOS FUTUROS

Como trabalho futuro, há a possibilidade de utilizar as informações extraídas dos *postmortems* e, a partir das atividades realizadas, sugerir boas práticas de desenvolvimento para jogos. Dessa forma, é possível verificar as técnicas e práticas mais usadas, que deram certo e errado, mesclar com práticas existentes e também com recomendações dos próprios desenvolvedores. No entanto, para isso, é necessário extrair mais dados e aumentar a base de processos.

REFERÊNCIAS

ANDERSON, E. F. *et al.* Choosing the infrastructure for entertainment and serious computer games-a whiteroom benchmark for game engine selection. In: IEEE. *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2013 5th International Conference on*. [S.l.], 2013. p. 1–8. Citado na página 71.

ARMBRUST, O. *et al.* Scoping software process lines. *International Journal on Software Process: Improvement and Practice*, v. 14, n. 3, p. 181–197, 2009. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1002/spip.412/abstract>. Citado na página 29.

BATES, B. *Game Design*. Thomson Course Technology, 2004. ISBN 9781592004935. Disponível em: <https://books.google.com.br/books?id=f7XFJnGrb3UC>. Citado 2 vezes nas páginas 51 e 69.

BECK, K. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2000. (An Alan R. Apt Book Series). ISBN 9780201616415. Disponível em: <https://books.google.com.br/books?id=G8EL4H4vf7UC>. Citado na página 29.

BETHKE, E. *Game Development and Production*. Wordware Pub., 2003. (Wordware game developer's library). ISBN 9781556229510. Disponível em: <https://books.google.com.br/books?id=m5exIODbtqkC>. Citado na página 69.

BLOW, J. Game development: Harder than you think. *Queue*, ACM, v. 1, n. 10, p. 28, 2004. Citado na página 23.

BOEHM; TURNER, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. ISBN 0321186125. Citado 3 vezes nas páginas 31, 32 e 35.

CALLELE, D.; NEUFELD, E.; SCHNEIDER, K. Requirements engineering and the creative process in the video game industry. In: IEEE. *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*. [S.l.], 2005. p. 240–250. Citado 2 vezes nas páginas 23 e 32.

FOWLER, M. The new methodology. *Wuhan University Journal of Natural Sciences*, Springer, v. 6, n. 1-2, p. 12–24, 2001. Citado na página 30.

FRIDLEY, M. *Postmortem: Kingdoms of Amalur: Reckoning*. 2013. Disponível em: <http://www.gamasutra.com/view/feature/197269/postmortem{_}kingdoms{_}of{_}am>. Citado na página 32.

FUGGETTA, A. Software process: A roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM, 2000. (ICSE '00), p. 25–34. ISBN 1-58113-253-0. Disponível em: <http://doi.acm.org/10.1145/336512.336521>. Citado na página 29.

FUGGETTA, A.; NITTO, E. D. Software process. In: ACM. *Proceedings of the on Future of Software Engineering*. [S.l.], 2014. p. 1–12. Citado na página 31.

GERSHENFELD, A.; LOPARCO, M.; BARAJAS, C. *Game plan: the insider's guide to breaking in and succeeding in the computer and video game business*. New York: St. Martin' s Griffin Press, 2003. Citado na página 23.

GRIL, J. *The State of Indie Gaming - Gamasutra*. 2008. <http://www.gamasutra.com/view/feature/132041/the_state_of_indie_gaming.php>. Citado na página 70.

HAMANN, W. Goodbye postmortems, hello critical stage analysis. *Gamasutra The Art and Business of Making Games*, jul 2003. Citado na página 32.

HTMLTOPDF. *HTML to PDF*. 2017. [Online; accessed 22-Fev-2017]. Disponível em: <http://wkhtmltopdf.org/>. Citado na página 47.

HUMPHREY, W. S. The software engineering process: Definition and scope. In: *Proceedings of the 4th International Software Process Workshop on Representing and Enacting the Software Process*. New York, NY, USA: ACM, 1988. (ISPW '88), p. 82–83. ISBN 0-89791-314-0. Disponível em: <http://doi.acm.org/10.1145/75110.75122>. Citado 2 vezes nas páginas 28 e 35.

JOLLIFFE, I. *Principal component analysis*. [S.l.]: Wiley Online Library, 2002. Citado 3 vezes nas páginas 15, 33 e 34.

JR, M. W. *et al.* "what went right and what went wrong": An analysis of 155 postmortems from game development. In: *Proceedings of the 38th International Conference on Software Engineering (ICSE 2016 SEIP Track)*. ACM – Association for Computing Machinery, 2016. Disponível em: <http://research.microsoft.com/apps/pubs/default.aspx?id=262301>. Citado na página 38.

KANODE, C. M.; HADDAD, H. M. Software engineering challenges in game development. In: IEEE. *Information Technology: New Generations, 2009. ITNG'09. Sixth International Conference on*. [S.l.], 2009. p. 260–265. Citado 2 vezes nas páginas 23 e 38.

KEITH, C. *Agile Game Development with Scrum*. 1st. ed. [S.l.]: Addison-Wesley Professional, 2010. ISBN 0321618521, 9780321618528. Citado na página 23.

KRUCHTEN, P. Contextualizing agile software development. *Journal of Software: Evolution and Process*, Wiley Online Library, v. 25, n. 4, p. 351–361, 2013. Citado 4 vezes nas páginas 31, 32, 35 e 68.

LARMAN, C.; BASILI, V. R. Iterative and incremental development: A brief history. *Computer*, IEEE, n. 6, p. 47–56, 2003. Citado na página 30.

MCGUIRE, M.; JENKINS, O. *Creating Games: Mechanics, Content, and Technology*. Taylor & Francis, 2008. (Ak Peters Series). ISBN 9781568813059. Disponível em: <https://books.google.com.br/books?id=swvSgicJM5IC>. Citado na página 69.

MOORE, M.; NOVAK, J. *Game Development Essentials: Game industry career guide*. Delmar/Cengage Learning, 2010. (Game development essentials). ISBN 9781428376472. Disponível em: <https://books.google.com.br/books?id=D-spAQAAMAAJ>. Citado na página 51.

MURPHY-HILL, E.; ZIMMERMANN, T.; NAGAPPAN, N. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In: *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, 2014. Disponível em: <http://research.microsoft.com/apps/pubs/default.aspx?id=210047>. Citado 4 vezes nas páginas 23, 24, 37 e 38.

MUSIL, J. *et al. A Survey on the State of the Practice in Video Game Software Development*. [S.l.], 2010. Citado na página 30.

MYLLYAHO, M. *et al.* A review of small and large post-mortem analysis methods. In: 17TH INTERNATIONAL CONFERENCE SOFTWARE & SYSTEMS ENGINEERING AND THEIR APPLICATIONS. *IEEE France*. Paris, 2004. Citado na página 33.

NAUR, P.; RANDELL, B. (Ed.). *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*. [S.l.: s.n.], 1969. Citado na página 29.

NEWZOO. *THE GLOBAL GAMES MARKET REACHES $99.6 BILLION IN 2016, MOBILE GENERATING 37%*. 2016. <https://newzoo.com/insights/articles/global-games-market-reaches-99-6-billion-2016-mobile-generating-37/>. Citado na página 23.

O'HAGAN, A. O.; O'CONNOR, R. V. Systems, software and services process improvement: 22nd european conference, eurospi 2015, ankara, turkey, september 30 – october 2, 2015. proceedings. In: ____. Cham: Springer International Publishing, 2015. cap. Towards an Understanding of Game Software Development Processes: A Case Study, p. 3–16. ISBN 978-3-319-24647-5. Disponível em: <http://dx.doi.org/10.1007/978-3-319-24647-5_1>. Citado na página 38.

O'HAGAN, A. O.; COLEMAN, G.; O'CONNOR, R. V. Software development processes for games: a systematic literature review. In: *Systems, Software and Services Process Improvement*. [S.l.]: Springer, 2014. p. 182–193. Citado na página 38.

PARKER, L. *The Rise of the Indie Developer - Gamespot*. 2011. <http://www.gamespot.com/articles/the-rise-of-the-indie-developer/1100-6298425/>. Citado na página 70.

PETRILLO, F.; PIMENTA, M. Is agility out there?: agile practices in game development. In: ACM. *Proceedings of the 28th ACM International Conference on Design of Communication*. [S.l.], 2010. p. 9–15. Citado 4 vezes nas páginas 23, 30, 37 e 66.

PETRILLO, F. *et al.* Houston, we have a problem...: a survey of actual problems in computer games development. In: ACM. *Proceedings of the 2008 ACM symposium on Applied computing*. [S.l.], 2008. p. 707–711. Citado 2 vezes nas páginas 23 e 24.

PETRILLO, F. *et al.* What went wrong? a survey of problems in game development. *Computers in Entertainment (CIE)*, ACM, v. 7, n. 1, p. 13, 2009. Citado 2 vezes nas páginas 23 e 37.

POLITOWSKI, C. *et al.* Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry. In: *GAS 2016*. [S.l.: s.n.], 2016. ISBN 9781450341608. Citado 6 vezes nas páginas 24, 30, 39, 51, 58 e 64.

POLITOWSKI, C. *et al.* Software engineering processes in game development: a survey about brazilian developers' experiences. 2016. Citado 2 vezes nas páginas 39 e 64.

POPPENDIECK, M.; POPPENDIECK, T. *Lean Software Development: An Agile Toolkit*. Addison-Wesley, 2003. (The agile software development series). ISBN 9780321150783. Disponível em: <https://books.google.com.br/books?id=8o1eom6ifIMC>. Citado na página 30.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. [S.l.]: Palgrave Macmillan, 2005. Citado na página 30.

ROBILLARD, M.; WALKER, R.; ZIMMERMANN, T. Recommendation systems for software engineering. *IEEE software*, IEEE, v. 27, n. 4, p. 80–86, 2010. Citado na página 27.

ROBILLARD, M. P. *et al. Recommendation systems in software engineering*. [S.l.]: Springer Science & Business, 2014. Citado 3 vezes nas páginas 24, 27 e 28.

ROYCE, W. W. Managing the development of large software systems. In: LOS ANGELES. *proceedings of IEEE WESCON*. [S.l.], 1970. v. 26, n. 8, p. 328–388. Citado na página 30.

SCHWABER, K.; BEEDLE, M. *Agile Software Development with Scrum*. Pearson Education International, 2002. (Series in agile software development). ISBN 9780132074896. Disponível em: <https://books.google.com.br/books?id=lO3XLgAACAAJ>. Citado na página 30.

SOMMERVILLE, I. *Software Engineering*. Pearson, 2011. (International Computer Science Series). ISBN 9780137053469. Disponível em: <https://books.google.com.br/books?id=l0egcQAACAAJ>. Citado na página 29.

VERSIONONE. *The 10th State of Agile Report*. [S.l.], 2015. Citado na página 30.

WESTSTAR, J.; ANDREI-GEDJA, M. *Developer Satisfaction Survey 2015 Industry Trends and Future Outlook Report*. [S.l.], 2016. Citado na página 24.

WGET. *GNU Wget*. 2017. [Online; accessed 22-Fev-2017]. Disponível em: <https://www.gnu.org/software/wget/>. Citado na página 47.

WIKIPEDIA. *Crowdfunding — Wikipedia, The Free Encyclopedia*. 2016. [Online; accessed 3-October-2016]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Markdown&oldid=736891244>. Citado na página 49.

WIKIPEDIA. *Crowdfunding — Wikipedia, The Free Encyclopedia*. 2016. [Online; accessed 1-July-2016]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Crowdfunding&oldid=727827466>. Citado na página 70.

WIKIPEDIA. *Web scraping — Wikipedia, The Free Encyclopedia*. 2016. [Online; accessed 1-July-2016]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=727574979>. Citado na página 47.

Apêndices

APÊNDICE A – FORMULÁRIO DE VALIDAÇÃO DO PROCESSO EXTRAÍDO DO *POSTMORTEM*

Feedback about EXTRACTED process

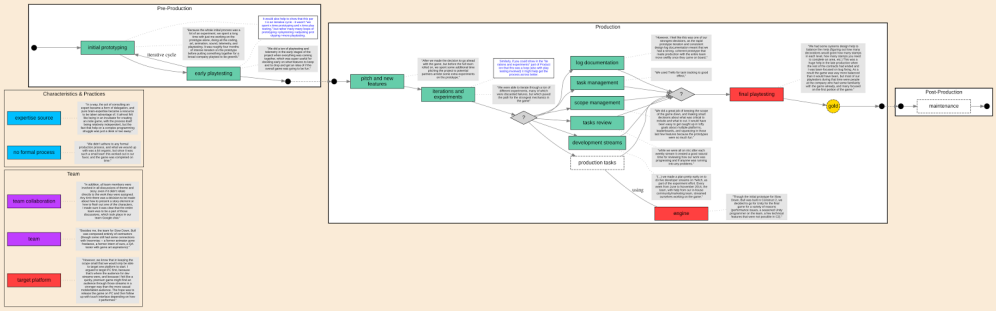
Hi Mr./Ms. <POSTMORTEM AUTHOR>

I'm a Brazilian master's degree student, working with games and software engineering. Part of my research is extract the processes used in game development projects and, as a source, I'm using postmortems. Although it is not the postmortem's purpose, it's the best source I have.

So, I read your postmortem about <GAME> and tried figure out the development workflow. I would be happy if you take five minutes of your time to examine the process I drew from the analysis:

Thank you very much, this is a great help in my research. If you have any doubt, please contact me.

Process extracted for the game "Slow down, bull"



1. **Roughly, is this workflow similar of what you used when developing your game?**

2. **What are the elements that does not make sense?**

   .....

3. **Will you add something important?**

   .....

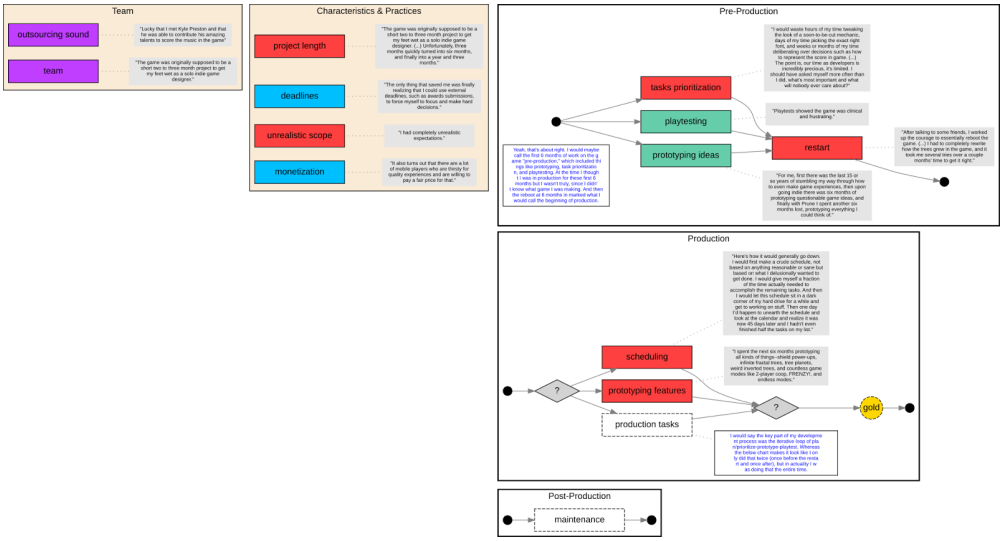
APÊNDICE B – FORMULÁRIO DE VALIDAÇÃO DO PROCESSO RECOMENDADO PELO SISTEMA

Feedback about RECOMMENDED process

Hi Mr. <DEVELOPER AND POSTMORTEM AUTHOR>. I'm the Brazilian guy who is researching about game development process from postmortems. We exchanged some emails a while back and now I really need your help once more to validate my dissertation results. It won't take more than 5 or 10 minutes. Once again, thank you very much for your time.

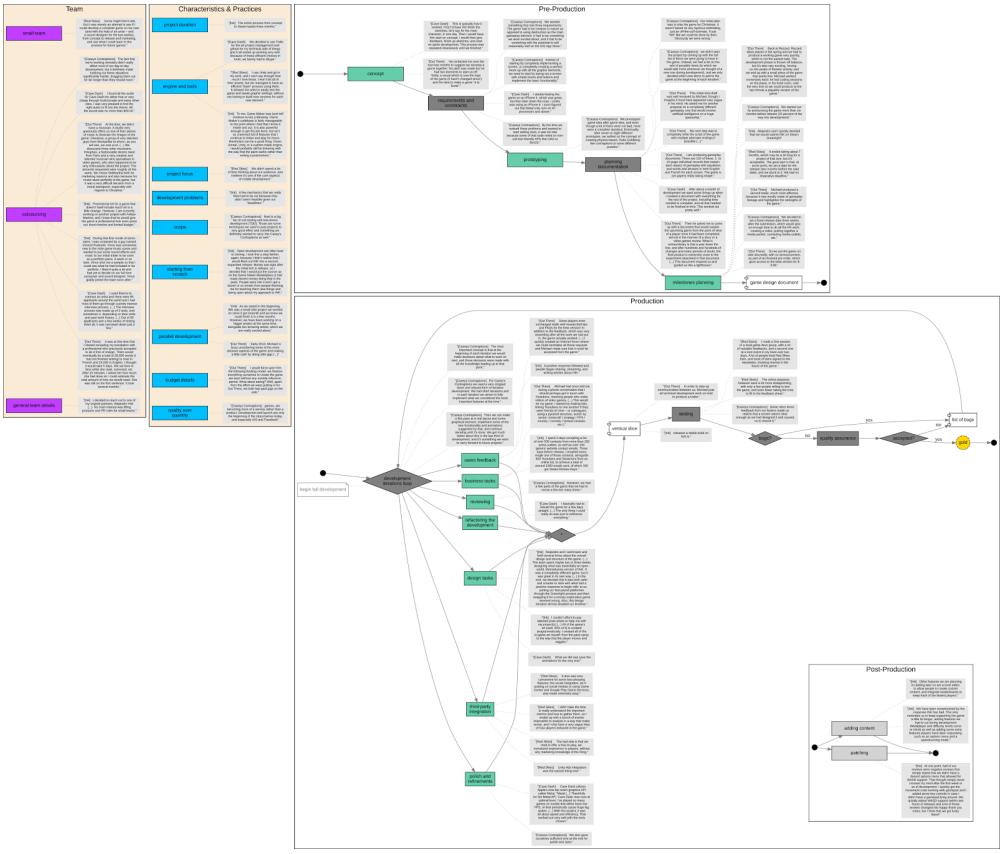
First, as you may remember, I extracted the development process from your postmortem about the project <GANE>.

<GAME> process extracted from postmortem



So, we developed a tool that find the most similar projects and merge the activities, resulting in a new process. We use the <GAME> project as input. The similar projects were the following games: <LIST OF SIMILAR GAMES>. The result was the following process:

Recommended process for <GAME> context



Considering the new process, please answer the following questions:

1. **In general, is this new workflow similar to what you used developing the game?**

2. **Which process elements does not make sense?**

3. **Do you add something important to this workflow?**

4. **If you began a new video game project similar to GAME, what would be the feasibility of using this new process?**
Mark only one oval.

	1	2	3	4	5	
unfeasible	◯	◯	◯	◯	◯	more feasible

APÊNDICE C – FORMULÁRIO ENTRADA PARA O CONTEXTO DO JOGO

Video Game Context Form

This is the form which the project context will be extracted.

There are 5 sections about Activities, Team, Project Management, Technical and Design.

Activities context

Information regarding development process activities.

1. **In what way do you want to develop the game:**
   Mark only one oval.

   ◯ I intend to develop the game incrementally, adding complete functionality throughout development, embracing changes and additions of new features.
   ◯ I intend to advance planning all (or most) of the functionality, document and strictly follow the plan.

2. **Will be used features prototyping to validate an idea before its conception?**
   Mark only one oval.

   ◯ Yes
   ◯ No

3. **The game performance will have a proper stage or team focused on it?**
   Mark only one oval.

   ◯ Yes
   ◯ No

4. **Will the engine and / or other help tools be built by the developer team itself?**
   Mark only one oval.

   ◯ Yes
   ◯ No

5. **The team intends to buy ready assets from thirds?**
   Mark only one oval.

   ◯ Yes
   ◯ No

6. **Will the team be composed by outsourced members?**
   Mark only one oval.

   ◯ Yes
   ◯ No

7. **About the pre-production phase:**
   Mark only one oval.

   ◯ The team intends to go straight to production, ignoring the pre-production stage and leaving validations for another phase.
   ◯ The team prefers to work in an exploratory phase, creating prototypes and testing hypotheses in search of the "fun" factor.

8. **Is it expected to add post-release content as well as bug fixes and lots of patches to balance the game?**
   Mark only one oval.

   - ◯ Yes
   - ◯ No

9. **Are there any components or any tool that can be reused for this project?**
   Mark only one oval.

   - ◯ Yes
   - ◯ No

10. **About testing and QA:**
    Check all that apply.

    - ☐ The team has a team of tests and / or QA
    - ☐ Feedbacks will be taken by users in a closed beta
    - ☐ Feedbacks will be taken by users in an open beta
    - ☐ Feedbacks will be taken by users in an early access version

11. **Tarefas relacionadas a marketing serão feitas pelo próprio time de desenvolvimento? Who will do the marketing and business tasks?**
    Mark only one oval.

    - ◯ The development team
    - ◯ A third party

Team context

12. **What is the estimated team size?**
    Mark only one oval.

    - ◯ Less than 5
    - ◯ between 5 and 25
    - ◯ Bigger than 25

13. **The team is divided in two or more teams?**
    Mark only one oval.

    - ◯ Yes
    - ◯ No

14. **The team members work remotely?**
    Mark only one oval.

    - ◯ Yes
    - ◯ No

Project Management context

15. **What is your developer type?**
    Mark only one oval.

    - ◯ First Party
    - ◯ Second Party
    - ◯ Third Party

16. **Are ou a Indie game developer?**
    What is indie?
    Mark only one oval.

    - ◯ Yes
    - ◯ No

17. **About the funding:**
    Mark only one oval.

    - ◯ External
    - ◯ Self
    - ◯ Crowdfunding

18. **About the Publisher:**
    Mark only one oval.

    - ◯ We are the publishers
    - ◯ He is external

Technical context

19. **What type of intellectual property is the game?**
    Mark only one oval.

    - ◯ Port
    - ◯ Remaster / Reboot
    - ◯ Franchise Sequence
    - ◯ Expansion
    - ◯ Mod
    - ◯ New IP

20. **About the Engine:**
    Mark only one oval.

    - ◯ In-House (new):
    - ◯ In-House Ready:
    - ◯ Off-The-Shelf:

Platform context

21. **What are the target platforms?**
    Check all that apply.

    - ☐ Console: Microsoft
    - ☐ Console: Sony:
    - ☐ Console: Nintendo:
    - ☐ Pc: Windows:
    - ☐ Pc: Mac:
    - ☐ Pc: Linux:
    - ☐ Mobile: Android:
    - ☐ Mobile: Ios:

Design context

22. **What is the game genre?**
Mark only one oval.

- Action:
- Action-Adventure:
- Adventure:
- Role-Playing:
- Simulation:
- Strategy:
- Puzzle:
- Sports:

23. **What are the game modes?**
Check all that apply.

- [ ] Single-Player:
- [ ] Multi-Player (offline):
- [ ] Multi-Player Online:

APÊNDICE D – TABELA COM VALORES DE CONTEXTO

	ACTIVITIES																				TEAM				
# Project	Agile	Prototyping	Performance Optimization	Tools Development	Outsourcing: Assets	Outsourcing: Work	Pre-Production: Short	Pre-Production: Long	Post-Production: Normal	Post-Production: Heavy	Reuse: Code	Reuse: Assets	Testing: In-Ouse Qa Team	Testing: Closed Beta	Testing: Open Beta	Testing: Early Access	Marketing/Pr: Self	Marketing/Pr: Outsourced	Size: <=5	Size: 5 – 25	Size: >25	Type: Single	Type: Collaborative	Distribuited
1	✓	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
2	✗	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
3	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓
4	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✓	✓
5	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗
6	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗
7	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
8	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
9	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✓	✓	✗	✗	✓	✗	✗	✓	✓
10	✓	✓	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓
11	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓
12	✗	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
13	✗	✓	✗	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
14	✗	✗	✗	✗	✓	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
15	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
16	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗
17	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
18	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗	✗
19	✗	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
20	✓	✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✓
21	✗	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✓	✗	✓
22	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
23	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗
24	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
25	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
26	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
27	✗	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
28	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓
29	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✓	✗
30	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓
31	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
32	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓
33	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓
34	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✓
35	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
36	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
37	✓	✓	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
38	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✓
39	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗
40	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓
41	✓	✓	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
42	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗
43	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗
44	✓	✓	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗
45	✗	✓	✗	✗	✓	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
46	✗	✓	✗	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓
47	✗	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗
48	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
49	✗	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
50	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
51	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗
52	✗	✓	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓
53	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗
54	✗	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗
55	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗

	PROJECT MANAGEMENT											TECHNICAL								
#	Project	Developer Type: First-Party	Developer Type: Second-Party	Developer Type: Third-Party	Indie	Funding: External	Funding: Self	Funding: Crowdfunding	Publisher: External	Publisher: Self (same Developer)	Intelectual Property: Port	Intelectual Property: Remaster / Reboot	Intelectual Property: Franchise Sequence	Intelectual Property: Expansion	Intelectual Property: Mod	Intelectual Property: New Ip	Engine: In-House (new)	Engine: In-House Ready	Engine: Off-The-Shelf
1		✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
2		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
3		✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
4		✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗
5		✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
6		✗	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
7		✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
8		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
9		✗	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
10		✗	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
11		✗	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓
12		✗	✗	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓
13		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
14		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
15		✓	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
16		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓
17		✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
18		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗
19		✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
20		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗
21		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
22		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
23		✗	✗	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓
24		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗
25		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
26		✗	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
27		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
28		✗	✗	✓	✓	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
29		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
30		✗	✗	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓	✗	✓	✗
31		✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✗
32		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓
33		✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗
34		✗	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
35		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
36		✗	✗	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
37		✗	✗	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗
38		✗	✗	✓	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
39		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
40		✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
41		✗	✗	✓	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓
42		✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓
43		✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
44		✗	✗	✓	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓
45		✗	✗	✓	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
46		✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
47		✗	✗	✓	✓	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
48		✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗
49		✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
50		✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗
51		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
52		✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗
53		✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗
54		✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗
55		✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✓	✓	✗	✗

	PLATFORM									DESIGN									
# Project	Console: Microsoft	Console: Sony	Console: Nintendo	Pc: Windows	Pc: Mac	[Pc: Linux	Mobile: Android	Mobile: Ios	Genre: Action	Genre: Action-Adventure	Genre: Adventure	Genre: Role-Playing	Genre: Simulation	Genre: Strategy	Genre: Puzzle	Genre: Sports	Mode: Single-Player	Mode: Multi-Player (offline)	Mode: Multi-Player Online
1	✓	✓	✘	✓	✓	✘	✓	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘
2	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✓
3	✘	✘	✘	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
4	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
5	✘	✘	✘	✓	✓	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✓	✘	✓
6	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✓	✘	✘
7	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
8	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘
9	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘
10	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
11	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
12	✓	✓	✓	✓	✓	✓	✘	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
13	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
14	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘
15	✓	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
16	✘	✘	✘	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
17	✓	✓	✓	✓	✓	✓	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓	✓	✘
18	✓	✓	✘	✓	✓	✓	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
19	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘
20	✓	✓	✘	✓	✘	✘	✘	✘	✓	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘
21	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
22	✘	✘	✘	✓	✓	✓	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
23	✘	✘	✘	✓	✓	✓	✓	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
24	✓	✓	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓
25	✘	✘	✘	✓	✓	✘	✓	✓	✘	✓	✘	✓	✘	✘	✘	✘	✓	✘	✘
26	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘
27	✘	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
28	✘	✘	✘	✓	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✓	✘	✘
29	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✓	✘	✓
30	✘	✘	✘	✓	✓	✘	✓	✓	✘	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘
31	✓	✓	✘	✓	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✓	✘	✓
32	✘	✓	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✓	✘	✓	✘	✘
33	✘	✘	✘	✘	✘	✘	✓	✓	✘	✓	✘	✘	✘	✘	✓	✘	✓	✘	✘
34	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘
35	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘
36	✓	✓	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘	✘	✓	✘	✘	✓	✓	✘
37	✓	✓	✘	✓	✘	✘	✘	✘	✓	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘
38	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
39	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✓	✓	✓
40	✘	✘	✘	✓	✓	✓	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
41	✘	✘	✘	✘	✘	✘	✓	✓	✘	✘	✘	✘	✘	✓	✘	✘	✓	✓	✓
42	✘	✘	✘	✓	✓	✓	✓	✓	✘	✘	✘	✓	✘	✘	✘	✘	✓	✓	✓
43	✓	✓	✓	✓	✓	✓	✘	✘	✓	✘	✘	✘	✘	✘	✓	✘	✓	✓	✓
44	✘	✘	✘	✓	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✓	✓
45	✘	✘	✘	✓	✓	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
46	✓	✓	✘	✓	✓	✓	✘	✓	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
47	✓	✓	✘	✓	✓	✓	✓	✓	✘	✘	✘	✘	✘	✓	✘	✘	✓	✘	✘
48	✘	✘	✘	✓	✘	✘	✓	✓	✘	✘	✘	✘	✓	✘	✘	✘	✓	✘	✘
49	✘	✓	✓	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘
50	✓	✓	✓	✓	✓	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
51	✓	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✓	✓
52	✘	✓	✘	✓	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✘	✘
53	✘	✓	✘	✘	✘	✘	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✓	✓
54	✘	✓	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✘	✓	✘	✓	✘	✘
55	✓	✓	✓	✓	✓	✓	✘	✘	✘	✓	✘	✘	✘	✘	✘	✘	✓	✓	✓

Anexos

ANEXO A – EXEMPLO DE *POSTMORTEM*

An indie-style experiment at a AAA studio: Insomniac's Slow Down, Bull

gamasutra.com/view/news/259163/An_indiestyle_experiment_at_a_AAA_studio_Insomniacs_Slow_Down_Bull.php

This postmortem, written by current indie and former triple-A dev Lisa Brown tells the story of the development of Insomniac's* Slow Down, Bull *-- an indie-style small game made by a big, well-known developer.

Insomniac Games has a reputation for always being willing to experiment. Whether it's trying to blend game genres, evolving a proven gameplay mechanic or branching out into a new platform, that spirit is something I've admired for a long time.

In the summer of 2013, mid-production on *Sunset Overdrive*, we tried a different kind of experiment, and I was thrilled to be involved. The premise: How far could one person take a prototype before needing to roll a team onto the game? Could we also make a great game with a small team and shorter timeline than our typical big budget console games?

When building the prototype for the pitch that ultimately became *Slow Down, Bull*, I started with a few mechanics constraints. First, I wanted to make a game with constrained input, only two buttons. Second, I wanted to try a game where your input stopped movement instead of caused it.

Eventually, this prototype turned into Insomniac's first small PC game and first foray into the realm of open development. *Slow Down, Bull* is an action collecting game about a stressed out, overachiever bull named Esteban who just wants to collect beautiful things, but is constantly worried that he isn't doing well enough. It became a charming little game wherein we partnered with Starlight Children's Foundation to give roughly half the net proceeds to the charity.

It was definitely a bit of a wild experiment for us in a number of ways, and we learned many things along the way.

What Went Right

1. Long prototyping phase

Because the whole initial process was a bit of an experiment, we spent a long time with just me working on the prototype alone, doing all the coding, art, animation, sound, telemetry, and playtesting. It was roughly four months of intense iteration on the prototype before putting something together for a broad company playtest to be greenlit.

After we made the decision to go ahead with the game, but before the full team rolled on, we spent some additional time pitching the project to potential partners amidst some extra experiments on the prototype. Do note that this wasn't a continuous timeline (the studio hibernates for a brief time during the winter holidays), but even still it may seem like a long time to stew on a single prototype.

However, I feel like this was one of our strongest decisions, as the rapid prototype iteration and consistent design log documentation meant that we had a strong, coherent prototype that made production with the entire team move swiftly once they came on board. We were able to iterate through a ton of different experiments, many of which were discarded failures, but which paved the path for the strongest mechanics in the game (the bullcatcher, the possum, and even the cat all were birthed out of a long line of experiments.)

Some of the discarded prototypes included a red light/green light mode, a mode in which you had to collect pickups in predetermined order, a pickup that increased your stress the longer you held it, a mode where you had to steer on a specific path, and a thief who stole decorations that you had to charge into. While these were ditched for not being particularly fun, they helped clarify what WAS fun and distinct about the steering and stress

mechanics, and thus the time spent on trying them was incredibly valuable.

2. Developer Streams

When *Slow Down, Bull* was greenlit, we made a plan pretty early on to do live developer streams on Twitch, as part of the experiment effort. Every week from June to November 2014, the team, with help from our in-house community/marketing team, streamed ourselves working on the game.

Sometimes it was me building a level, or the programmer implementing an enemy behavior, or the artist creating environment assets, or our composer creating a song. During the streams the team also answered audience questions about game development and generally chatted about process.

I feel like the streams were really positive on several fronts. It unified the team under a common weekly event (streaming yourself building the game is pretty stressful, so it bonded us together!) It also connected us with a curious audience of fans. Some of our favorite moments were letting our stream audience come up with names for the characters in the game, or meeting people at events like PAX who had been coming to the streams since the beginning, and who expressed how grateful they were for being shown “behind the curtain” of the development process.

Production-wise, doing the weekly streams actually made the team work in a very efficient way. We felt accountable to show progress to our stream audience every week, and so were calculated in how we scheduled and scoped the game. The fact that the streams proved successful on a production pipeline front was a bit of a surprise, since they were a fair amount of work to wrangle every week.

3. Team ownership

Besides me, the team for *Slow Down, Bull* was composed entirely of contractors (though some still had some connections with Insomniac -- a former animator gone freelance, a former intern of ours, a QA tester with game art aspirations). Since the game ended up being a highly personal one for me in theme, I knew that it was going to be important that the team felt a connection to the game beyond just being hired hands to produce it.

On the other hand, by the time the team came on there was a strong vision for the feeling of the game, and on a relatively short production cycle (about seven months) we had to be able to hit the ground running. In the end, I felt like we were really successful in creating a strong team bond in which everyone felt creative ownership over the game, which yielded each component of work feeling very personal.

Part of this was finding ways to get the team members to relate to the overarching theme of the game (an overachiever who is stressed out by that which he loves, always worrying that it isn't good enough). Besides being a common struggle of highly creative people, the developer streams ended up bringing everyone together over this theme, as we were all passionate about showing the development process but nervous about showing our unfinished, imperfect work to the public. It made us all relate to the story of Esteban, and was easier to put a bit of ourselves into the game.

In addition, all team members were involved in all discussions of theme and story, even if it didn't relate directly to the work they were assigned. Any time there was a decision to be made about how to present a story element or how to flesh out one of the characters, I made sure it was clear that the entire team was to be a part of those discussions, which took place in our team Google chat.

4. Access to internal resources

The biggest advantage that the large studio setting gave this project was access to resources. During the prototyping phase, other designers were frequently involved in the playtesting process. Studio leads were available to help with questions and processes for leading the project (for example, preparing strong pitches, advice on wrangling contractors, etc.) When we ran into a tricky animation problem, we had a veritable army of highly skilled, experienced animators to consult.

In a way, the act of consulting an expert became a form of delegation, and pure brain-expertise became a resource to be taken advantage of. It almost felt like being in an incubator for creating an indie game, with the process itself being relatively independent, but the fact that help on a complex programming struggle was just a desk or two away.

At one point in the project when thinking about feedback on the collectibles, I polled the entire company on what made juicy feeling pickups, and what made the bolts in *Ratchet & Clank* feel so good. Every department had a different take, of course, and years of experience on that particular piece of polish. Compiling that sort of data was invaluable, and I think a big help on a project with such a small team.

5. Good scoping and production management

We did a great job of keeping the scope of the game down, and making smart decisions about what was critical to include and what to cut. It would have been easy to get caught up in lofty goals about multiple platforms, leaderboards, and squeezing in those last few features because the prototypes were so much fun.

But I think we had a pretty strict, systematic way of addressing cuts, looking clearly at how many resources it would actually take to get things done, and proposing them in such a way that features and cuts would get approved (sometimes it surprises people to learn that even cuts can need approval, and that a cut in a game still creates work, so making those decisions early in our production was important).

I also felt like our team worked efficiently, encouraged by things like our developer streams and also our means of constant communication. Two of our team members were remote and in different time zones, so we made it a habit of always being available in our team Google Hangout, with periodic calls with the whole team. We used Trello for task tracking to good effect, and while we were all on mic after each weekly stream it created a good natural time for reviewing how our work was progressing and if anyone was running into any problems.

We didn't adhere to any formal production process, and what we wound up with was a bit organic, but since it was such a small team this worked out in our favor, and the game was completed on time.

What Went Wrong

1. Not good late telemetry (balancing was off)

We did a ton of playtesting and telemetry in the early stages of the project when everything was coming together, which was super useful for deciding early on what features to keep and to drop and get an idea of if the overall

game was going to be fun. However, towards the end of the project, we got caught under the timeline of finishing the game on time and didn't devote as much time and resources to telemetry and playtesting.

We had some systems design help to balance the meta (figuring out how many decorations would grant how many stamps in each level, how many stamps you need to complete an area, etc.) This was a huge help in the late production when the rest of the contracts had ended and I was laser-focused on bug fixing. As a result the game was way more balanced than it would have been, but most of our playtesters during that time were people at the company who had some familiarity with the game already, and many focused on the first portion of the game.

The result is that the first couple of areas balance out pretty well, but then you get difficulty spikes around Area 3, and the last area of the game has too lax of a stamp requirement to beat. The progression pacing is just off. Doing more thorough pre-release testing and telemetry with external playtesters would have helped iron out the balancing.

2. Streaming and market

Though the developer streams were something I considered a big "what went right" for the project, they came with their own set of mistakes. *Slow Down, Bull* has a very simple control scheme, and in initial prototypes we tested in mouse, controller, and touch. However, we know that in keeping the scope small that we would only be able to target one platform to start. I argued to target PC first, because that's where the audience for dev streams were, and because I felt like a quirky, premium game might find an audience through those streams in a stronger way than the more casual mobile/tablet audience. The hope was to release the game on PC and then follow up with touch interface depending on how it performed.

Though we had a small, consistent group of enthusiastic regulars in the streams, we didn't gain a huge following in spite of pushing the streams regularly to the Insomniac following and weren't able to secure a Twitch front-page push. There were struggles in dealing with how to leverage the streams as a growth tool, much of which was due to some timing challenges (addressed below), but in the end our findings pointed more to the fact that the market was not there after all. Choosing a mobile distribution first, or even a different PC distribution besides or in addition to Steam that might have reached a different audience (itch.io, Humble, etc.) would probably have been a better approach.

Looking back on it, some of it is the usual amount of luck, as it's impossible to tell when a concept is going to

take off, but a big part is being more aggressive about finding your audience early. Showing the game around in various contexts, I found that many people who it resonated with weren't necessarily traditional gamers (for example, traditional artists who loved the look), or they were gamers who spoke specifically about the refreshing nature of the nonviolent approach the game took, or folks who were once overachieving students in liberal arts education who identified with the character.

These were people I encountered on an individual basis when showing the game in informal settings, but finding them earlier would have given better insight about distribution. Do these people play games on PC? If they do, do they get them through Steam? Do they play games on mobile? We could have done more in finding exactly where the audience for this game was, and determining whether or not it was big enough to be sustainable earlier.

Finding these people isn't the same as running a focus test, but rather finding them involves getting the game out there -- showing it at game festivals but perhaps non-games events as well, and talking about the game early in as many outlets as you can. This is why I think this piece of advice is so prevalent for indies -- doing this legwork to find the audience early can inform you about the best platform through which to reach them.

3. Leaderboards are a crutch

This is a lesson learned that I feel can be applied to any game of any size. In early prototyping, within a smaller group of testers who already know one another, leaderboards can be a crutch that draws attention away from what you need to be learning about your game. In the first company playtest of the prototype, I hacked in some rough leaderboards for high scores on each level to see how that felt. The result was that a small number of hyper competitive people had a great time trying to one-up each other for the high scores for each level.

While this was great fun, I think it gave a false sense of where the fun came from in the game relative to the amount of work it takes to actually implement leaderboards in a game "for reals." When we had a final understanding of our timeline and I decided to cut leaderboards up front so that we could actually finish the game on time, there was some response of confusion since, after all, I'd gotten leaderboards up and running within a couple of weeks for the prototype. (However, that "prototype" leaderboard was completely unsecure and barely reliable).

I was conflicted on whether to put this as a "what went wrong" at first because, after all, at the time of the prototype I didn't know if leaderboards could be a thing in the final game and wanted to test them out. However, I wish I'd done that initial playtest without them, so I could have gotten better insights into the fun of the mechanics themselves without the distraction of "it's fun to beat my co-worker to get the high score."

4. Unity's animator (not knowing when to roll our own solutions)

Though the initial prototype for *Slow Down, Bull* was built in Construct 2, we decided to go for Unity for the final game for a variety of reasons (performance issues, a seasoned Unity programmer on the team, a few technical features that were not possible in C2). We decided early on to make use of Unity's animator tool, rather than controlling how and when animations were played in code, mostly because a general look at what it was capable of seemed like it was usable.

The animator at the time seemed deceptively useful, but ended up causing us more headaches than anything else in production, and I feel like we should have done a bit more deep-dive testing of the system before committing to it. There were a lot of problems with using the animator for certain things that also had rotation or translation dependencies on code events (for example, Esteban's head was rotated in code in certain circumstances).

We also had a lot of trouble with bugs in more complicated state transitions between animations for certain characters, especially anything that involved any sort of layering (Annette the bullcatcher was notorious for not transitioning properly through her various tell states, even when all the variables were set, due to the unintuitive transition settings in the animators overriding what animation was being told to play in the state machine).

At the end of the project, it felt like we had had to fight and conquer Unity's animator to get it to do what we wanted, rather than it being a tool to help productivity. The more general takeaway here is that, while out-of-the-box tools are amazing in making small scale development fast and accessible, it is still important to take time to do a bit of trade-off analysis and decide if it's better for your project to roll your own tools or methods.

5. Making a little game in a big studio

I mentioned earlier that a big pro to doing a small scale production in a large studio was access to resources, but the big studio environment created some obstacles as well. One big challenge was melding Insomniac's triple-A console and mobile marketing experience with something more experimental in nature.

Slow Down, Bull is a small and quirky game that is incredibly personal to the team who made it, and typically with small indie games the personal slant can be leveraged in marketing the game. However, in a bigger studio with many projects, each game has to fit within the brand of the company and share promotion time with all of the other projects. It's a balancing act. This game had to be seen as an Insomniac game first and foremost, and though we highlighted some of the personal nature through the dev streams, pushing a development story that was very close to the individuals on the team would have been unfair to the rest of the company as a whole.

The end of production fell to bad timing, right during our push on *Sunset Overdrive* and then the Android port of *Outernauts*. It was important for the company to not divide its message across games, so the result was there was a big delay between wrapping up the developer streams and the launch of the game so we couldn't tie launch promotion to the streams. Many times smaller games tend to aggressively stir up word well before launch to pinpoint its audience well ahead of time.

We ended up doing something in between, with regular blog posts, dev streams, and press-driven announcement prior to launch, but with many other projects to manage, we didn't really have the bandwidth to do the aggressive, early audience-seeking promotion that you see in a lot of smaller niche titles.

We ran into a few issues with "small game at big studio" in other places, too, in looking to submit it to festivals. There was a disconnect between the small scrappy nature of development and the fact that it was a game from a large established studio, which meant that it didn't really "fit" in a lot of places that better serve smaller indie teams.

Lest I come off as bashing the marketing team, the reasons for resistance here are perfectly reasonable. When you have a large company with an already established brand, controlling that image and keeping it consistent is paramount, or else a lot of things can go wrong. I just think we weren't anticipating some of these challenges early on when we decided to do the experiment to begin with.

Conclusion

In the end, I feel like we made a solid, quirky game with an unusual mechanic and an incredibly charming, relatable story. The schoolcraft art is beautiful and the music is amazing. Watching streams and Let's Plays of the game and seeing people "get" how the stress of the gameplay resounds with the nature of the story and character has been really rewarding. However, we didn't find an audience large enough to be sustainable through the distribution channel that we chose, though in the end we gained a lot of insight about the challenges of making small games at a big studio.

Will Insomniac try more games like this in the future, changing things up with all the lessons we learned? That remains to be seen!

As for me, the whole experience has resulted in me going indie. It made me realize that to be at my best and happiest was to be making games for the joy of making them, and to be able to make games for smaller audiences that may not be sustainable to a large studio (even if the large studio is making a small game).

The developer streams were such a positive experience for me that I'm focusing most of my work on that -- streaming production on small, experimental game development, educating aspiring game devs and curious fans

- and being able to do so in a context that is more easily suited to smaller, personal games than within a large studio structure that still has to be concerned with market. The challenges of marketing on my own terms and building a community from scratch are ones I feel ready to face after the learning experience of this project.

Lisa Brown** was the designer and project lead for* **Slow Down, Bull, having previously been a designer on* **Resistance 3** *and* **Sunset Overdrive***. She has since made the indie plunge and is off having wild indie adventures, mostly involving streaming game development.*