Uma Proposta de Quadro Kanban Para Promover Visibilidade em Projetos de Jogos

Paulyne Matthews Jucá[1], **Arthur de Castro Callado**[1]

[1]Campus de Quixadá – Universidade Federal do Ceará (UFC)
Av. José de Freitas Queiroz, 5003 – 63.902-580 – Quixadá – CE – Brazil

{paulyne,arthur}@ufc.br

Abstract. *The game development process still presents many challenges. One of these challenges concerns improving project management, especially in factors related to the visibility of blocking states, the overall progress of the project, and controlling the impact of the appearance of unplanned work. This article proposes a new organization for Kanban boards used by game development studios to promote more visibility to situations faced in the daily life.* ***Keywords.*** *Kanban, Game development process, Game project management*

Resumo. *O processo de desenvolvimento de jogos ainda apresenta muitos desafios. Um desses desafios diz respeito melhorar o gerenciamento do projeto, especialmente em fatores relacionados à visibilidade dos estados de bloqueio, ao andamento geral do projeto e no controle do impacto do aparecimento de trabalhos não planejados. Esse artigo propõe uma nova organização para quadros Kanban usados por estudios de desenvolvimento de jogos para promover mais visibilidade às situações enfrentadas no cotidiano.* ***Palavras-chave.*** *Kanban, Processo de desenvolvimento de jogos, Gerenciamento de projeto de jogo*

1. Introdução

O desenvolvimento de jogos é uma área multidisciplinar que envolve esforço de artistas gráficos, desenvolvedores de software, músicos, animadores, designers, entre outros. Cada profissional usa um conjunto de técnicas e ferramentas da sua subárea para criar as partes que irão compor o jogo. Isso faz com que uma equipe de desenvolvimento de jogos seja composta de várias subequipes para tratar dessas partes específicas, especialmente para jogos maiores. As metodologias ágeis sugerem que equipes pequenas e auto-gerenciáveis criam produtos com capacidade de evolução e entrega mais constante. Assim, para um jogo pequeno, é comum ter uma equipe pequena mista com poucas pessoas assumindo sozinhas alguns dos papéis no desenvolvimento. Para jogos maiores, o mais comum é ter equipes pequenas de profissionais de uma especialidade específica cuidando de uma parte do problema completo. Em qualquer configuração, a necessidade de coordenar os trabalhos e comunicar decisões é um fator chave.

Em um trabalho de 2021 [Politowski et al. 2021], os autores levantaram os principais problemas ainda existentes no desenvolvimento de jogos. Alguns dos problemas identificados estão relacionados com falta de visibilidade geral das dependências do projeto, muito trabalho em progresso, falta de priorização de trabalho, entre outros.

Este trabalho tem foco em dar visibilidade aos estados de bloqueio, ao andamento geral do projeto e em controlar o impacto que o aparecimento de trabalhos não planejados causa ao desenvolvimento de jogos.

Para isso, tomou como base as recomendações em [DeGrandis 2017] para propor um quadro Kanban que ataque esses problemas citados.

O quadro Kanban proposto nesse trabalho ainda precisa passar por uma validação. Entretanto, o resultado esperado com esse quadro é a proposta de uma solução para um problema ainda muito comum no desenvolvimento de jogos. Mesmo que as empresas façam adaptações sobre o quadro proposto, apenas a reflexão sobre como dar visibilidade ao problema pode ajudar a promover um melhor ambiente para o desenvolvimento de jogos.

2. Trabalhos Relacionados

[DeGrandis 2017] debate os problemas de que roubam a produtividade de empresas de desenvolvimento de software e propõe mudanças no quadro Kanban usado por times de desenvolvimento de software para dar visibilidade a esses problemas. Ela sumariza recomendações e passos para proposição de quadros Kanban que podem ser usados por empresas de software para gerar quadros que atendam às suas necessidades. Esse trabalho utiliza os passos propostos por ela para sugerir um quadro Kanban específico para indústria de jogos. A principal diferença é que, apesar de [DeGrandis 2017] tratar dependências desconhecidas, na indústria de software as dependências são mais isoladas uma vez que é raro que designer e programadores, por exemplo, precisem mudar seus códigos em razão do resultado do trabalho do outro. Uma vez definidas interfaces claras, a interseção do trabalho é menor. Em jogos, o impacto do trabalho entre as áreas é muito maior. Um trabalho pode precisar ser totalmente refeito por falta de comunicação entre programadores e artistas, por exemplo. Assim, o quadro Kanban aqui proposto tem mais ênfase nas dependências entre áreas.

[Schmalz et al. 2014] faz uma análise do gerenciamento de risco em projetos de desenvolvimento de jogos. Em 2014, ele afirmava que apesar de existirem muitos trabalhos relacionados ao gerenciamento no desenvolvimento de software e que, embora o desenvolvimento de jogos tenham muitos desafios semelhantes, existiam poucos trabalhos relacionados especificamente ao gerenciamento no desenvolvimento de jogos.

[Martins et al. 2021] relatam oportunidades, dificuldades e experiências no processo de desenvolvimento de jogos casuais em uma parceria entre indústria e academia. Nesse trabalho, relatam impactos que a mudança de equipe, de escopo e de mecânicas acarretaram no desenvolvimento de dois jogos e como essas mudanças impactou a experiência dos desenvolvedores durante o projeto. O trabalho aqui proposto tem o foco maior em propor mudanças na estrutura do quadro Kanban para promover maior visibilidade sobre alguns dos problemas comuns em desenvolvimento de jogos. A relação com o trabalho de [Martins et al. 2021] é que eles também encontraram alguns dos problemas de bloqueio e falta de visibilidade nos projetos analisados por eles.

Muitos trabalhos como [Belarmino et al. 2021], [Mangeli et al. 2021] e [Keith 2010] propuseram adaptações ágeis para o desenvolvimento de jogos. Esses trabalho estão relacionados a esse por tratarem do processo de desenvolvimento de jogos. Entretanto esse trabalho tem o foco não no processo como um todo, mas em parte do processo, especialmente a atividade de gerenciamento de projeto.

3. Quadro Kanban

"Kanban não é um sistema de controle de estoque, mas pode ser considerado um sistema de visualização do trabalho, tornando-o fluido, reduzindo o desperdício e maximizando o atendimento do valor para o cliente. Ele controla o fluxo de produção passando a demanda do cliente por toda a cadeia de produção" (tradução livre de [Corona and Pani 2012]).

Segundo [Corona and Pani 2012] e [DeGrandis 2017], o quadro Kanban é a principal ferramenta usada para visualizar e coordenar o trabalho em equipe que usam processos ágeis. As colunas do quadro mostram a sequência de etapas do processo de desenvolvimento. Os cartões representam as atividades a serem realizadas. Para cada etapa, há limites para o número de atividades simultâneas daquela etapa. Isso limita o volume de trabalho em andamento evitando que uma equipe comece muita coisa, mas não termine nada. Trabalho em andamento cria atividades zumbis que são atividades que não terminam, mas ficam poluindo o quadro e usando tempo de acompanhamento e gerando preocupação na equipe. Ainda segundo [Corona and Pani 2012] e [DeGrandis 2017], o quadro pode também ter linhas (chamadas de swinlines) que podem indicar prioridades diferentes de atividades, versões diferentes de entrega ou organização de partes do time. "Outras variantes do quadro Kanban incluem quadros com linhas adicionais, representando diferentes projetos, com faixa de emergência para urgências, com zonas segurando cartões representando bugs ou problemas em aberto" [Corona and Pani 2012]. Nesse trabalho, as linhas irão representar prioridades das atividades, incluindo linha para urgência, atividades com ou sem dependência e melhorias opcionais, mas desejadas pelo time.

4. Problemas no Desenvolvimento de Jogos

[Politowski et al. 2021] fez um levantamento sobre os problemas identificados através da análise de *postmortems* no desenvolvimento de jogos. Eles agruparam os problemas em 3 categorias: produção, gerenciamento e negócios. Os problemas relatados na produção incluem problemas com o *gamedesign* e balanceamento do jogo, documentação, problemas com as ferramentas, com os testes, presença de bugs e falta de prototipação. Os problemas de negócio estão relacionados à propaganda e maneiras de faturamento do jogo. Essas duas categorias estão fora do escopo desse trabalho.

A outra categoria está relacionada ao gerenciamento do projeto. Os problemas encontrados aqui incluem escopo não-realista com muitas funcionalidades, adição de funcionalidades não planejadas, atraso de entregas, excesso de trabalho (*crunch*), problemas de comunicação, dificuldade em contratação, custos maiores que os previstos, trabalhos e projetos paralelos, muito/pouco tempo de planejamento e vazamento de materiais. Muitos desses problemas estão relacionados especificamente a gerenciamento de tempo.

[DeGrandis 2017] explica como atividades não planejadas ou não priorizadas ou escondidas podem roubar tempo e, com isso, dinheiro dos projetos. Os problemas também afetam a qualidade de vida da equipe. A sessão seguinte detalha o trabalho de [DeGrandis 2017] sobre como dar visibilidade às atividades que roubam o tempo e a produtividade de equipes de desenvolvimento.

5. Ladrões de tempo

No livro "Making Work Visible" [DeGrandis 2017], a autora aponta 5 ladrões de tempo no desenvolvimento de software e sugere formas de definir quadros Kanban que deem

visibilidade a esses problemas. Os cinco principais ladrões de tempo são 1) muito trabalho em andamento 2) dependências desconhecidas 3) trabalho não planejado 4) conflitos de prioridade 5) trabalho negligenciado.

Ela [DeGrandis 2017] trata no livro como esses problemas atrapalham a produtividade das equipes e gera atrasos no desenvolvimento do projeto. Como solução, a autora propõe que dar visibilidade aos problemas ao propor melhorias na definição dos quadros Kanban das equipes. Para permitir o aprendizado das sugestões ela [DeGrandis 2017] propõe uma série de exercícios. Esses exercícios não estão formalmente organizados como uma metodologia, mas podem ser seguidos como passos de uma metodologia e é isso que esse trabalho segue.

A autora [DeGrandis 2017] propõe começar definindo o vocabulário e as atividades mais comuns da equipe. Essas atividades podem ser diferentes para cada subequipe, mas ela sugere que se agrupe ao máximo possível atividades semelhantes para evitar a introdução de complexidade desnecessária. As etapas do processo viram colunas no quadro. Os diferentes tipos de atividades viram cores diferentes de "cards". Essa parte não chega a ser muita novidade na elaboração do quadro e todas as equipes que personalizam quadros fazem isso instintivamente.

Em seguida, a autora [DeGrandis 2017] sugere deixar as atividades que tem dependência de outras mais visíveis no quadro. Ela sugere algumas formas de dar visibilidade, sendo uma delas a criação de uma trilha própria para atividades que tem dependência. O quadro assim fica muito diferente do que geralmente é usado pois ganha uma novas dimensão com linhas que trazem significado adicional ao agrupamento de atividades.

Como terceiro passo, ela [DeGrandis 2017] sugere a explicitação das atividades não planejadas. Ela dá exemplos de 2 tipos de atividades não planejadas : as interrupções e as atividades urgentes de última hora. As interrupções atrapalham o andamento do projeto, pois obrigam a troca de contexto e a perda de concentração e do fluxo de trabalho. Elas fazem a atividade levar mais tempo para ficar pronta, mas geralmente não estão contabilizadas no quadro. Isso causa uma sensação ele improdutividade na equipe que não percebe como o tempo foi gasto. Ela sugere a adição de uma coluna para contabilizar interrupções no quadro. Ela faz a mesma sugestão de inclusão do uma linha para atividades urgentes. Nesse caso, a linha também tem o propósito de dar visibilidade às atividades priorizadas pela urgência. Por fim, a autora sugere deixar tempo no cronograma alocando espaços de tempo para essas atividades não planejadas, se elas forem muito comuns ao projeto.

O próximo passo é definir claramente a regra de priorização de atividades [DeGrandis 2017]. Isso fica claro no quadro com a adição de uma coluna antes da coluna que lista as atividades a serem feitas agora (backlog geral). A ideia é separar as atividades que a equipe se comprometeu a desenvolver, das atividades que podem no futuro ser desenvolvidas.

O próximo passo é dar visibilidade às atividades que foram negligenciadas [DeGrandis 2017]. Uma forma de fazer isso é marcar atividades que não evoluem no quadro há muito tempo. Se a atividade é importante, deve ser tratada. Se ela deixou de ser importante, deve ser abortada. Atividades zumbi roubam tempo do projeto.

6. Metodologia

A metodologia proposta por este trabalho está descrita a seguir:

1. Identificar quais os problemas mais comuns no desenvolvimento de jogos;
2. Selecionar os que têm relação com falta de visibilidade e gestão de tempo;
3. Utilizar as técnicas propostas por [DeGrandis 2017] para criar um quadro Kanban para resolver os problemas selecionados e verificar as dependências entre as atividades;
4. Discutir o quadro Kanban proposto.

A ideia de verificar a dependência entre as atividades advém do fato de que em projetos ágeis de desenvolvimento de software em geral há poucas dependências entre as atividades, portanto não é tratado em detalhes por [DeGrandis 2017]. Entretanto, isso não ocorre em projetos de desenvolvimento de jogos, onde a quantidade de dependências é tipicamente grande e pode levar a estados de bloqueio com uma equipe esperando que outra termine seu trabalho para poder continuar. Assim, a grande quantidade de dependências impacta o quadro Kanban proposto que recebe várias linhas específicas para tratar o problema (atividades com e sem dependência e atividades em bloqueio).

7. Identificando os Problemas Relacionado ao Roubo de Tempo

Avaliando a lista de problemas identificados por [Politowski et al. 2021], é possível perceber que os problemas listados como problemas de gerenciamento são os mais interessantes para o escopo desse trabalho. Assim, os problemas que essa proposta pretende tratar são:

1) Escopo não realístico com muitas funcionalidades: esse problema pode estar relacionado a três dos "ladrões de tempo": muito trabalho em progresso, dependência não clara, falta de prioridade e trabalho negligenciado. Quando não existe um controle sobre a quantidade de atividades simultâneas que o profissional pode iniciar, é possível que muitas atividades sejam iniciadas e não finalizadas, aumentando muito a quantidade de trabalho em progresso. Em jogos, devido às muitas dependências de uma atividade, é comum a equipe decidir "adiantar trabalho". Essa prática ajuda na diminuição do impacto negativo de dependências, mas pode gerar muitas atividades paradas à espera de ação por outros equipes. Assim, é importante que o trabalho a ser adiantado faça parte das funcionalidades que foram priorizadas. Quando o trabalho adiantado não faz parte das prioridades, ele vira zumbi no quadro, contribuindo para aumentar a quantidade de trabalho em andamento e afetando a produtividade da equipe. O trabalho iniciado pode até estar terminado para parte da equipe que o adiantou, mas representa trabalho negligenciado e pendência para a equipe como em todo.

2) Adição de novas funcionalidades não planejadas: esse problema tem ligação direta com o ladrão de tempo do mesmo nome, mas também pode ter relação com a falta do priorização e com a geração de muitos trabalhos em paralelo. A chegada de novas funcionalidades por si só não representa um problema. Ela vira um problema quando se acumula com o trabalho em andamento gerando um compromisso da equipe com a funcionalidade e mudança de prioridade das atividades em andamento. Quando atividades em andamento precisam parar para permitir a execução dessas novas funcionalidades, então as atividades antes em andamento podem passar muito tempo negligenciadas ou virando zumbis. Se o projeto precisa adotar novas funcionalidades de forma rotineira,

deve fazer uma alocação mais folgada de atividades planejadas para permitir espaço para tratar a chegada de funcionalidades não esperadas.

3) Problemas de comunicação: esse problema tem relação direta com muito trabalho simultâneo, dependências desconhecidas, muito trabalho não planejado e problemas de priorização. Fazer jogo é uma atividade criativa e multidisciplinar. Isso exige a coordenação de trabalhos de naturezas diferentes que se integram e retroalimentam. Entretanto, é comum que uma subequipe que tem autonomia para decisões no seu domínio de atuação (ex: modelagem, animação, programação) não ter visibilidade, ou mesmo interesse, nas atividades de responsabilidade de outra subequipe. A falta de visibilidade geral e de comunicação pode causar estados de bloqueio onde uma subequipe ou atividade fica impossibilitada de progredir. A identificação prévia e clara desses atividades, bem como sua priorização, pode antecipar problemas e reduzir problemas do comunicação.

4) Atrasos de entregas: tem relação com o aparecimento de funcionalidades não planejadas, muitas funcionalidades e problemas de comunicação. Por isso, tem relação com todos os ladrões de tempo já tratados. A mesma coisa vale para os custos quando decorrem do atraso de entregas.

5) Excesso de trabalho - Crunch: esse problema é muito decorrente dos problemas com funcionalidades com a adição do efeito das interrupções, estados de bloqueio e retrabalho. Interrupções fazem o trabalho andar mais lentamente e geram acúmulo de trabalho. As dependências geram estados de bloqueio quando se espera por informações ou ações de outras pessoas. Para não "ficar parado" o profissional inicia nova atividade e aumenta a quantidade de trabalho em progresso/andamento. Quando a atividade que esperava retorno depois de passar algum tempo em negligência retorna, o profissional precisem relembrar seu escopo e muitas vezes refazer parte do trabalho. Esse trabalho não planejando aumento a necessidade de horas extras para evitar o atraso do projeto se tornando em grave problema para todos os envolvidos no projeto.

6) Planejamento: o planejamento de um projeto de jogos precisa prever tempo para resolução de estados de bloqueio gerado pelas dependências, tempo para a chegada de atividades não planejadas ou urgentes e deve deixar claras as prioridades do projeto.

8. Proposta de Quadro Kanban para Jogos

O primeiro passo sugerido pela [DeGrandis 2017] é definir os passos do seu processo.

8.1. Passos para Definir o Quadro

Definir as linhas do quadro. Elas vão representar estados no desenvolvimento do jogo. Podem ser estados anteriores ao início do desenvolvimento em si, como o planejamento. Ou podem ser etapas de execução (desenvolvimento) de equipes diferentes que combinam suas atividades para gerar um resultado final, como a movimentação de um personagem que pode envolver atividades de animação, arte, programação e som que precisam ser integradas depois de desenvolvidas individualmente e que possuem dependências entre si. As colunas definidas foram:

1) ToDo (a fazer): atividades que foram priorizadas e esperam iniciar a produção. Elas não contam como trabalho em andamento e a coluna não tem limite de atividades. Pode ser entendida como a fila de atividades a serem feitas. É esperado que elas sejam

priorizadas levando em consideração a "história de usuário" ou fase do jogo a que elas pertencem.

2) Preparação: Nessa fase, o executor da atividade junta as informações necessárias para começar a sua execução propriamente dita, estudar o que for necessário e planejar a execução. A partir dessa fase, as atividades já começaram a ser trabalhadas e devem ser contadas como trabalho em andamento. As atividades nessa etapa podem ter dependências externas ou internas e ficarem em estado de espera ou bloqueio. É importante dar visibilidade às atividades que estão bloqueadas para evitar a criação de atividades zumbis ou criar urgências desnecessárias.

3) Execução: essa é a fase da construção do artefato do jogo. Para um programador pode ser código, para um designer pode ser um modelo 3D, por exemplo. Representa um trabalho feito por uma subequipe. Entretanto, nessa fase o executor pode precisar confirmar dados ou validar informações com outras subequipes. Essas dependências têm o potencial de bloquear a execução e deve ficar claro para todos os envolvidos quando os estados de espera ou bloqueio acontecem. No quadro proposto, a execução tem subcolunas representando os times de desenvolvimento (lilás), arte (verde), design (azul) e som (vermelho).

4) Integração: as atividades que foram feitas por uma subequipe só podem ser consideradas terminadas quando estiverem integradas no jogo final. Entretanto, não é incomum que algum retrabalho precise acontecer em decorrência da integração. Assim, atividades em estado de integração são atividades que estão esperando a execução de outras para serem consideradas completas.

5) Teste: quando um conjunto de atividades que representam uma história de usuário ou fase do jogo está integrada ao jogo, elas podem ser testadas. Atividades em teste podem falhar e precisar de retrabalho. Aqui as atividades não são mais entendidas como coisas isoladas, mas como um conjunto de atividades.

6) Feito: é o estado do conjunto de atividades pronto, integrado e testado no jogo. Opcionalmente, é possível adicionar o estado Feito-Feito para indicar um conjunto de atividades que estão feitas e já disponíveis em uma versão liberada (disponível para jogadores finais) do jogo. Este trabalho não vai considerar Feito-Feito.

O próximo passo é dar visibilidade às dependências das atividades.

8.2. Dar Visibilidade às Dependências

Para deixar claro que uma atividade tem dependências, foi sugerida uma linha nova (swimlane). Todas as atividades que possuem dependência estão nessa linha. É provável que a maioria das atividades tenha alguma dependência. Para isso, a sugestão é adicionalmente marcar no cartão da atividade usando cores quais os tipos de dependência de uma atividade.

No exemplo abaixo (Figura 2), o cartão com linha lilás indica uma atividade de software que possui dependência com a área de arte (verde).

Se o cartão estiver na trilha de dependência (ver Figura 3), mas fora da marcação de bloqueio, significa que a atividade tem dependência, mas no momento não precisa de interação com ninguém da área de arte.

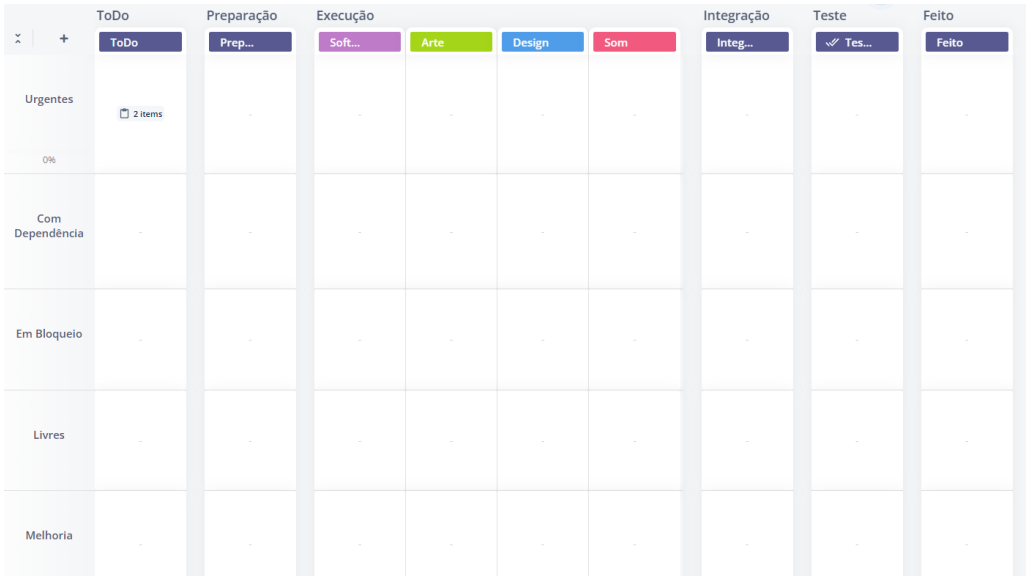

Figura 1. Quadro Kanban Proposto

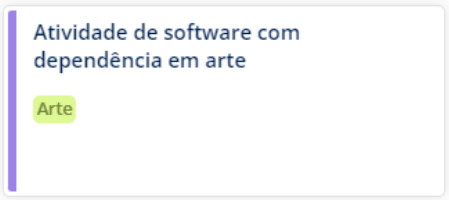

Figura 2. Exemplo de cartão com a marcação de dependência

Para indicar a necessidade de articulação ou que a atividade entrou em estado de bloqueio, o desenvolvedor de software deve colocar o cartão dentro da área de bloqueio na coluna de arte (Figura 4). Assim, visualmente fica claro para a equipe de arte que uma atividade de software está esperando por um retorno dela para continuar. Se essa espera se estender demais e a negociação virar urgente, o cartão deve subir para a linha de urgente na coluna de arte.

Essa solução ajuda tanto no problema de comunicação relacionado à visibilidade geral do projeto, pois permite que todos tenham ideia das atividades que irão depender da sua área (mesmo que não sejam originalmente da área), quanto ajuda a indicar prioridade na execução de atividades com dependência, uma vez que atividades em bloqueio param o fluxo de execução de uma pessoa ou uma subequipe.

A criação da coluna em preparação permite que as subequipes entendam se vale a pena começar uma atividade que depende de outra que vai levar mais tempo, mas nem começou a ser pensada. A sugestão, nesse caso, é adiar o início da atividade até que as dependências que precisam ficar prontas antes sejam iniciadas. Imagine que uma atividade de desenvolvimento dependa de decisões a serem tomadas na fase de design que nem começaram a ser discutidas. Não vale a pena iniciar o desenvolvimento, nesse caso.

A limitação da quantidade de atividades simultâneas por coluna no Kanban ajuda a evitar que a subequipe comece muitas coisas ao mesmo tempo e não conclua. Entretanto,

Figura 3. Exemplo de atividade com a marcação de dependência

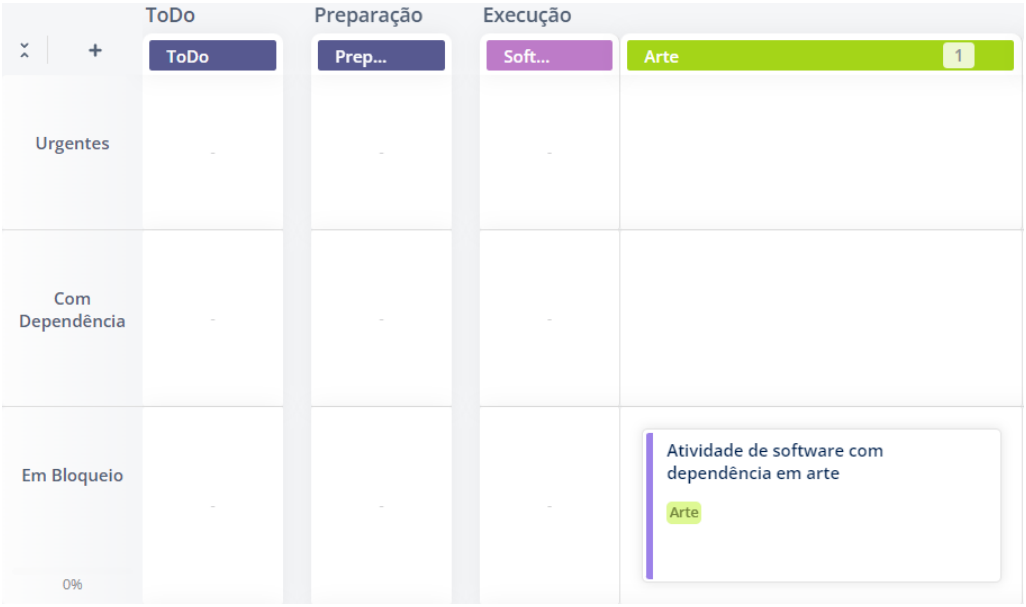

Figura 4. Exemplo de atividade com a marcação de bloqueio

o esforço de execução disponível para equipe de arte pode ser diferente do disponível para a equipe de software, mesmo que temporariamente. É por isso que os valores de execução são definidos de maneira individual para cada subequipe.

Quando uma subequipe estiver livre e não puder iniciar alguma atividade que tenha dependência, ele pode seguir o fluxo escolhendo atividades livre que não tem dependências.

A trilha de melhorias vai receber as atividades que chegaram ao nível feito, mas que, se tiver tempo no projeto, serão melhoradas e aprimoradas. Assim, o jogo pode melhorar com o tempo.

Atividades urgentes que não foram planejadas ou que foram negligenciadas por muito tempo e precisam terminar logo devem ser alocadas na trilha de urgência. Essas são as atividades eleitas como mais prioritárias.

Atividades que foram negligenciadas por muito tempo e foram canceladas podem ser guardadas em uma caixa adicional de canceladas. Essas atividades podem ajudar a manter a memória do projeto. Quando muitas atividades forem para a caixa de canceladas isso pode indicar que muito esforço do projeto está sendo gasto prematuramente e sem planejamento, especialmente se essas atividades não forem de game design.

O quadro Kanban proposto aqui agrupa todas as sugestões debatidas nesse trabalho. Entretanto, o resultado é um quadro mais complexo. Algumas equipes podem considerar o resultado final desvantajoso pela adição de complexidade. Esse trabalho não deve então ser percebido como uma exigência da adoção completa do quadro. Uma equipe que sentir que é interrompida por muito trabalho não planejado que possui alta prioridade pode apenas adotar a linha de urgência. Uma equipe que sente que as atividades ficam paradas muito tempo no quadro depois de começadas pode escolher dar visibilidade das dependências separando as atividades em livres (não dependentes) ou dependentes. Uma equipe que sinta que precisa deixar mais claro ainda os estados de bloqueio pode adotar adicionalmente ou isoladamente a linha de bloqueio. Uma equipe que gostaria de sinalizar o desejo de ter tempo para melhorar características do jogo que já tem uma versão pronta e testada, mas que sentem que podem fazer um trabalho mais refinado, podem adotar a linha melhoria para indicar/solicitar a priorização dessas tarefas, mas mantendo a distinção delas das outras tarefas que ainda não têm uma versão pronta.

Assim, não existe regra para a adoção do quadro. Ele pode ser entendido como conjuntos de soluções para problemas específicos que podem ou não ser adotados em conjunto.

9. Conclusões e Trabalhos Futuros

[Politowski et al. 2021] levantaram os principais problemas ainda existentes no desenvolvimento de jogos. Alguns dos problemas identificados estão relacionados com falta de visibilidade geral das dependências do projeto, muito trabalho em progresso, falta de priorização de trabalho, entre outros. Este trabalho tem foco em dar visibilidade aos estados de bloqueio, ao andamento geral do projeto e em controlar o impacto que o aparecimento de trabalhos não planejados causa ao desenvolvimento de jogos. Para isso, esse trabalho propõe um modelo de quadro Kanban como uma proposta de uma solução de alguns desses problemas. O quadro foi proposto usando as recomendações e os passos de criação de quadros Kanban proposto por [DeGrandis 2017] e adicionando o tratamento específico de dependências de jogos. Como resultado, foram sugeridas adições de colunas e linhas para tratar problemas específicos relacionados ao desenvolvimento de jogos. O quadro final apresentado é o somatório dessas sugestões. O quadro proposto pode ser adotado integralmente ou parcialmente pelos estúdios de jogos que identificarem alguns dos problemas tratados no seu dia-a-dia. O quadro Kanban proposto nesse trabalho ainda precisa passar por uma validação com o objetivo de explicitar, por exemplo, para cada tipo de jogo ou estúdio, quais das adições propostas têm maior impacto e quais têm menor impacto. Entretanto, um dos resultados desse trabalho é uma proposta de como dar visibilidade aos estados de bloqueio, ao andamento geral do projeto e em controlar o impacto que o aparecimento de trabalhos não planejados causa no desenvolvimento de jogos.

É esperado que a discussão aqui proposta também ajude desenvolvedores, pro-

dutores e gestores de estúdios de jogos a ter um ponto de partida na resolução desses mesmos problemas, promovendo um melhor ambiente para o desenvolvimento de jogos.

O principal trabalho futuro é a validação da proposta. Para esta validação, será necessário utilizar o quadro proposto integralmente na produção de pelo menos um jogo por um estúdio de jogos. Ao final da criação do jogo, espera-se, através de entrevista com o gerente da equipe, a obtenção de dados qualitativos sobre o impacto da adoção do quadro na produção do jogo, especialmente a melhoria na solução dos mesmos com uma visualização mais explícita das suas ocorrências, com a visibilidade das dependências proporcionada, por exemplo, pela coluna de integração proposta neste trabalho.

Referências

Belarmino, G., Oliveira, R., Rodriguez, C., Goya, D., and Rocha, R. (2021). Descrição e análise dos processos de produção de um jogo educacional e seus impactos na sua qualidade. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 407–416, Porto Alegre, RS, Brasil. SBC.

Corona, E. and Pani, F. E. (2012). An investigation of approaches to set up a kanban board, and of tools to manage it. In *Proceedings of the 11th International Conference on Telecommunications and Informatics, Proceedings of the 11th International Conference on Signal Processing*, SITE'12, page 53–58, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

DeGrandis, D. (2017). *Making Work Visible: Exposing Time Theft to Optimize Work & Flow*. IT Revolution Press.

Keith, C. (2010). *Agile Game Development with Scrum*. A Mike Cohn signature book. Addison-Wesley.

Mangeli, E., de Classe, T., Macêdo, H., Marques, P., Costa, L., and Xexéo, G. (2021). Metodologia para desenvolvimento de jogos com propósito de um laboratório de ludologia. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 143–151, Porto Alegre, RS, Brasil. SBC.

Martins, L., Carneiro, N., Miranda, D., Aquino, F., Castro, R., Andrade, R., and Darin, T. (2021). Interação entre academia e indústria no processo de desenvolvimento de jogos: Percepções e lições aprendidas. In *Anais Estendidos do XX Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 779–785, Porto Alegre, RS, Brasil. SBC.

Politowski, C., Petrillo, F., Ullmann, G. C., and Guéhéneuc, Y.-G. (2021). Game industry problems: An extensive analysis of the gray literature. *Information and Software Technology*, 134:106538.

Schmalz, M., Finn, A., and Taylor, H. (2014). Risk management in video game development projects. In *2014 47th Hawaii International Conference on System Sciences*, pages 4325–4334.