Desenvolvimento Orientado a Modelos de Jogos Sérios: processo e *design* com um jogo do gênero aventura e *quiz*

Diego Martos Buoro
Pós-Graduação em Ciência da Computação
Universidade Federal do ABC - UFABC
Santo André, Brasil
diego.martos@ufabc.edu.br

Rafaela Vilela da Rocha
Centro de Matemática, Computação e Cognição da UFABC
Santo André, Brasil
rafaela.rocha@ufabc.edu.br

Denise Hideko Goya
Centro de Matemática, Computação e Cognição da UFABC
Santo André, Brasil
denise.goya@ufabc.edu.br

***Resumo*—Por causa da capacidade de imersão e engajamento, jogos sérios ganharam popularidade para o treinamento educacional e profissional. O desenvolvimento de jogos orientado a modelo (MDGD) se mostra uma alternativa promissora para a criação de jogos sérios por não especialistas em programação, como professores e *designers*, ao reduzir as barreiras técnicas de programação existentes para a introdução do conteúdo pedagógico. Entretanto, ainda falta uma visão unificada e sistematizada do MDGD para a sua aplicação com relação a diferentes jogabilidades e temas, considerando também os artefatos produzidos e pessoas envolvidas. Para solucionar esse problema, este trabalho propõe um processo sistematizado do MDGD, com fases e atividades definidas, para aplicar em jogos sérios com qualquer gênero de jogabilidade. Para comprovar sua utilidade, apresenta-se uma linguagem específica de domínio (DSL) em desenvolvimento para jogos sérios do gênero *quiz* e aventura. Como resultados já obtidos, são apresentadas as fases de "Análise" e "Design" com os gêneros propostos. Como trabalhos futuros, serão implementados os artefatos propostos e realizada a validação da DSL e do jogo criado.**

***Palavras-chave*—Jogos Sérios, Desenvolvimento de Jogos Orientado a Modelo, Linguagem Específica de Domínio**

I. Introdução

Jogos Sérios (JS) são desenvolvidos com propósitos além do entretenimento, pois acrescentam objetivos pedagógicos e de avaliação e, em geral, são aplicados com sucesso em educação, simulação e treinamento [1], [2]. Os JSs são capazes de proporcionar o desenvolvimento de competências conforme os objetivos estabelecidos, que incluem habilidades cognitivas, motoras e sociais [3]–[6]. Dentro dos diferentes gêneros de jogabilidade, têm-se *quiz* e aventura. Quizzes são uma forma divertida e engajadora de aprendizagem baseada em jogos digitais, ao responder uma sequência perguntas para obter uma pontuação e *feedback* imediato [7]. O gênero aventura permite ao jogador explorar um mundo virtual, interagir com personagens, administrar recursos e ramificações da história [8].

Devido às peculiaridades dos JSs, principalmente a necessidade de ter uma boa jogabilidade e experiência com o conteúdo pedagógico, é necessária a adoção de metodologias e estratégias que atinjam esses objetivos [9], [10]. A complexidade de produção do jogo sério aumenta com a inclusão do especialista de domínio (professor, pedagogo, treinador, entre outros), que é responsável pelo conteúdo "sério", ou seja, as informações didáticas apresentadas no jogo. Com o surgimento de novas tecnologias, o especialista pode atuar diretamente no *design* do JS com a introdução de ferramentas de autoria [11], [12]. Além disso, há trabalhos que defendem a importância do envolvimento direto do especialista de domínio ou *designer* instrucional no processo de produção de um JS, sendo necessário um canal de comunicação com os desenvolvedores [13], [14].

Entretanto, as ferramentas de autoria não exploraram de forma integral os diferentes contextos, pois são dependentes de recursos pré-fixados, como cenários, imagens e personagens, sem a possibilidade de adicionar criações próprias [15]. Outro problema é que a simplicidade de ações pré-definidas pode impactar negativamente na experiência e engajamento do usuário [16]. Essas limitações refletem a necessidade de uma linguagem comum para uma comunicação efetiva, livre de barreiras técnicas de programação e do domínio de aplicação para a produção de JSs [14], [17], e que não restrinjam a criatividade do especialista.

Os problemas apresentados podem ser contornados pela metodologia de desenvolvimento orientado a modelos [15]. O desenvolvimento de jogos orientado a modelos (*Model-Driven Game Deve-lopment* - MDGD), como próprio nome indica, define que o projeto ou *design* do jogo é baseado em modelos de alta abstração que descrevem os elementos básicos e suas relações, sendo usados para a construção do jogo final por meio de ferramentas de automatização [18]. Outra característica é a presença de uma linguagem específica de domínio (*Domain-Specific Language* - DSL), com a qual é possível realizar a mudança do conteúdo do jogo, por meio de um alto grau de abstração. Portanto, nesse contexto, o especialista é capaz de alterar elementos do jogo com maior expressividade por meio de uma *interface* e, assim, gerar uma experiência personalizada [19]. Entretanto, ainda há barreiras na praticidade do MDGD, já que a maioria dos trabalhos na literatura apresenta um escopo de domínio real menor que o afirmado [18]. Além disso, por se tratar de um

assunto emergente, ainda não há estratégias bem definidas no processo de automatização e construção da DSL, por causa da variedade de gêneros de jogos e ambientes virtuais (2D ou 3D). Portanto, embora seja uma tecnologia ideal para reduzir os custos de produção de jogos sérios que envolvam participação de especialistas, até este momento é um desafio diminuir a complexidade na pré-produção dessa tecnologia.

Embora existam *frameworks* que visam integrar o desenvolvimento de jogos sérios com o MDGD, ainda não exploram em profundidade os objetivos pedagógicos [20] e foram projetados para gêneros de jogabilidades específicos [21], [22]. Ademais, não foram encontrados trabalhos que proponham processos que integram modelagem orientada a modelos, *design* de jogos, práticas de engenharia de software, e conteúdo e objetivos pedagógicos.

Desse modo, este trabalho propõe um processo para o desenvolvimento de DSLs para jogos sérios, que engloba atores, papéis, artefatos (ferramentas de edição, de transformação e DSL) e fases com atividades bem definidas. Para alcançar esse objetivo, foram definidas fases e atividades baseadas na literatura existente para *software*, mas com adaptações para adequar o processo ao desenvolvimento de jogos sérios e diferentes tipos de jogabilidade. Esse novo processo foi usado para a "Análise" e o "*Design*" de uma DSL focada no desenvolvimento de jogos sérios do gênero *quiz* e aventura.

O artigo está organizado como segue: na Seção II são apresentados os conceitos necessários para se compreender a proposta de DSL e MDGD em JSs. Na Seção III há um levantamento e discussão sobre metodologias de desenvolvimento de jogos e trabalhos que empregam MDGD em jogos dos gêneros *quiz* e aventura. Na Seção IV é proposto um processo sistematizado para apoiar a criação e modelagem de jogos sérios com orientação a modelos. Na Seção V é apresentada a Análise e o *design* da DSL para jogos de aventura e *quiz*, usando o processo proposto. Na Seção VI, discute-se a proposta, seguida de considerações finais.

II. Fundamentos em Jogos e Modelagem com DSL

Nesta seção são apresentados os principais conceitos relacionados a jogos digitais (gêneros aventura e *quiz*), jogos sérios (objetivos e desafios), desenvolvimento de jogos orientado a modelo e linguagem específica de domínio.

A. Jogos Digitais e Jogos Sérios

Jogos digitais podem ser categorizados em gêneros e há uma grande quantidade de gêneros por causa das diferentes ideias originais que surgiram ao longo da evolução da indústria de jogos [23] [24]. Dentre os que já foram explorados com interesse de pesquisa e são usados com sucesso em contexto educacional, estão os gêneros aventura e *quiz* [25]. Nos jogos de aventura, os principais elementos são exploração e resolução de quebra-cabeças, com uma história envolvente durante o jogo [26]. No tipo *quiz*, o jogador deve responder uma sequência de perguntas para obter um resultado final [7].

Dada a natureza textual e exploratória, um jogo de aventura foi definido como uma série de ramificações textuais que levariam jogadores a diferentes estados por meio de uma aventura [27]. Cavallari et al. [26], com base nas concepções de diferentes pesquisadores sobre o gênero, definiram, de maneira mais ampla, como um *software* que apresenta um ambiente virtual em um mundo imaginário no qual o usuário deve tomar decisões para resolver corretamente os problemas apresentados no jogo. Os elementos presentes são explorados pelo jogador ao interagir com personagens e objetos para completar tarefas, a fim de alcançar o objetivo final do jogo [8].

Nos *quizzes*, cada questão pode usar recursos audiovisuais para complementar ou enriquecer o texto e evitar a perda de interesse no jogo. Esse gênero oferece flexibilidade tanto na decisão de *design* de jogabilidade como também na representação e edição de conteúdo, e se tornou relevante para ser aplicado em jogos sérios [28]. Portanto, editar o conteúdo, seja incluindo, excluindo ou atualizando questões, é intuitivo dado o formato simples desse tipo de jogo. Também há vantagens para avaliação: é possível mapear tipos de erros por aprendiz [29] e os resultados podem ser usados para medir o nível de aprendizado, o que permite uma avaliação individualizada ou comparações entre aprendizes [30].

Jogo sérios, do inglês *serious games*, são jogos que vão além do entretenimento e objetivam a aprendizagem ou treinamento, avaliação e *feedback* imediato [31]. Os principais objetivos de um JS são: (1) ser desafiante e engajador e (2) garantir a educação, treinamento e avaliação em um dado contexto. Dessa forma, os objetivos pedagógicos devem ser estabelecidos desde a pré-produção do jogo, que envolve os requisitos necessários para o *design* do JS. Essa característica não está presente em jogos para entretenimento, embora ainda possam ser usados para fins didáticos [32]. Além disso, os jogos sérios podem se distinguir quanto ao domínio aplicado (defesa, educação, negócios, política, saúde, entre outros) e ao público-alvo (geral, estudantes, aprendizes e profissionais). Portanto, o escopo está atrelado com o propósito definido durante a fase pré-produção. Interesse e motivação são aspectos indispensáveis para que o aprendizado seja efetivo, e os jogos se tornaram uma boa ferramenta que auxilia o especialista de domínio no engajamento do conteúdo com os aprendizes [33].

Apesar dos efeitos positivos dos JS, ainda há desafios na produção destes por diferentes fatores [34]. O primeiro está nos requisitos e características do produto criado, que abrange aspectos como o domínio de aplicação, as regras de jogabilidade, a *interface* do usuário e *feedback* [35]. Além disso, é necessário atender aos princípios de aprendizagem afetiva, que inclui medir (competências, problemas e objetivos) e avaliar os resultados, e fornecer o *feedback* apropriado [36], [37]. Outro desafio é relacionado ao reúso e extensão de JS e artefatos, pois mudanças na aplicação implicam em alterações na arquitetura que são complexas e custosas [38].

B. Desenvolvimento de jogos orientado a modelo e DSL

Na área de desenvolvimento de *software*, o desenvolvimento de jogos orientado a modelos (MDGD) permite a transformação dos artefatos, como diferentes modelos, para a geração automática do jogo final [18], [39]. Essa abordagem

faz a combinação de geração de códigos de programação, linguagens específicas de domínio e transformações de artefatos. Os modelos representam artefatos primários, e estão diretamente ligados com a jogabilidade e os recursos usados no jogo. Uma vez que são modelos com alta abstração, são usados para gerar o *software* final, ou gerar outro modelo, e representam o *design* ou projeto do jogo. O processo de geração do jogo é feito por meio de ferramentas de automação, que são essenciais para encapsular o processo de *build* do *software*, o que elimina a necessidade de conhecimento de detalhes das tecnologias. Assim, uma ferramenta realiza o mapeamento do modelo conceitual para arquivos semiestruturados (artefatos de representação física ou instanciada do jogo) que são transformados por outra ferramenta para produzir o *software* final.

A característica marcante dessa metodologia é a presença de uma linguagem específica de domínio (DSL) que é uma linguagem de programação para uma determinada classe de problemas [40]. Assim, quanto mais específicas são as tarefas no contexto de aplicação, mais adequado é o uso da DSL para a solução dessas tarefas [41], [42]. O sucesso de algumas DSLs para aplicações mais tradicionais, como SQL para banco de dados e HTML para páginas *Web*, corroboram essa visão. Portanto, especificar o domínio representa um *tradeoff*, no qual a delimitação do escopo de problemas permite fazer uso de uma linguagem mais expressiva com anotações e abstrações, porém limita o manuseio dentro de um contexto.

Assim como linguagens gerais de programação, as DSLs contêm três características essenciais: a sintaxe concreta, a sintaxe abstrata e a semântica [43]. A sintaxe concreta, estruturada por meio da escrita de um dicionário, define a notação de como o usuário poderá se expressar nos programas, enquanto a sintaxe abstrata, também conhecida como metamodelo, refere-se à estrutura de dados usada para armazenar informações relevantes de maneira interligada. A semântica é o conjunto de restrições ou regras do sistema que o programa deve estar de acordo, além de estarem "corretos" com relação as sintaxes apresentadas anteriormente. Portanto, pode-se afirmar que é a semântica que definirá o que poderá ser manipulado, como se darão as relações entres elementos e o escopo da linguagem.

Faz-se necessário um compilador que realize tanto a análise sintática e léxica como também garanta a corretude realizada pela ferramenta de validação [44], [45]. Entretanto, fazer a construção dessas ferramentas por conta própria demanda conhecimento técnico elevado de programação. Além disso, garantir o funcionamento de todo o processo (análise sintática, análise semântica e otimizações antes da geração) implica um investimento maior de tempo gasto no início do desenvolvimento [46]. No caso das DSLs, para evitar esse esforço a mais de encadeamento de ferramentas e focar nas regras de transformação, seleciona-se um padrão que descreve a sintaxe abstrata ou metamodelo (como o *Meta-Object Facility* - MOF), pela qual o programador manterá foco apenas na construção e especificação da linguagem. Assim, não há a necessidade de implementar tanto o *parser* como o validador por conta própria, pois estão incluídos geradores de analisadores sintáticos. Portanto, é necessário apenas escrever as especificações da linguagem, que representarão os elementos e relações da linguagem, sendo o código gerado automaticamente. Essas especificações, que são um conjunto de regras que descrevem a forma de elementos que é válida de acordo com a sintaxe da linguagem, são chamadas de gramática [47].

Dessa forma, para garantir o funcionamento gramática da DSL, realiza-se a análise sintática ou *parse*, no qual será realizado o processamento de *tokens* que devem estar em conformidade com as regras da gramática. O resultado desse processo é a representação da estrutura em forma de árvore do código fonte do programa na memória, denominado de árvore sintática abstrata (AST, do inglês *abstract syntax tree*). Por último, dada a AST, é feita a análise semântica com o auxílio de uma tabela de símbolos, garantindo a corretude da linguagem ao verificar por erros que *parser* não consegue capturar, como a consistência entre tipos de dados e a garantia de um identificador único para todas as classes [48].

III. Trabalhos Relacionados

A. Metodologias de desenvolvimento de jogos sérios

Diferentes metodologias foram concebidas para auxiliar o processo de desenvolvimento de um JS, que envolve o estabelecimento de processos e conhecimento multidisciplinar para a construção do jogo. Aslan e Balci [49] criaram a metodologia *GAMED* (*diGital educAtional gaMe dEvelopment methoDology*), um ciclo de desenvolvimento iterativo de jogos educacionais nos quais as etapas são baseadas em métodos, regras e postulados. Rocha e colaboradores [50] propuseram o AIMED (*Agile, Interative and open Method for open Educactional resources Development*) que é um método ágil que integra práticas de *design* pedagógico, *design* de jogo, modelagem de simulação, engenharia de *software* e gerenciamento de projetos para auxiliar o desenvolvimento de recursos educacionais abertos, como jogos educacionais, jogos sérios, simulações interativas e artefatos gamificados. Algumas metodologias são focadas no tema de aplicação (educação, saúde, negócios, entre outros) e nos requisitos necessários e objetivos a serem alcançados para esses jogos específicos [51], [52]. Por outro lado, outros processos são focados no efeito de aprendizado ou treinamento obtido, baseando-se em métricas e valores quantitativos que estão ligados com o jogador [53].

B. Desenvolvimento da DSL em MDGD

Dadas as características do MDGD, é necessário ter passos elaborados para definir o domínio e a implementação da DSL. No contexto de metodologias para desenvolvimento de *software*, Van Deursen, Klint e Visse [46] categorizam as fases de elaboração da linguagem em três aspectos (análise, implementação e uso), com atividades bem definidas para auxiliar no *design*, que são: (1) identificar o domínio do problema; (2) coletar toda informação relevante sobre o domínio; (3) mapear essas informações em conjunto semântico de notações e operações; (4) projetar uma DSL que descreva concisamente as aplicações no domínio; (5) construir uma biblioteca que implemente as notações semânticas; (6) projetar

e implementar um compilador que traduza o programa de DSL produzido em chamadas de biblioteca; (7) escrever os programas com a DSL para as aplicações desejadas.

Os passos de (1) a (4) contemplam o aspecto de análise, ou seja, a necessidade de conhecer profundamente o domínio do problema, mapear como essas informações se relacionam para então projetar a DSL que realizará a construção do jogo. As atividades (5) e (6), que representam o aspecto de implementação, dizem respeito à construção da biblioteca e do artefato responsável pela transformação de código. Finalmente, o aspecto de uso cobre o passo (7), na qual ocorre a escrita de programas com a DSL. Entretanto, ainda falta clareza de quais artefatos devem ser gerados por causa das diferenças entre *softwares* e jogos digitais.

C. Criação de jogos do gênero aventura com MDGD

Laforcard e Laghouaouta [54] propuseram o desenvolvimento de cenários adaptativos que refletiram com a individualidade do aprendiz, por meio da MDGD. A geração de cenários é o resultado da transformação de modelos que representam o perfil do aprendiz e da descrição do jogo, o que facilita a criação de uma experiência única com imprevisibilidade. O jogo 2D do gênero *Escape-it* é representado com isometria para dar a sensação de profundidade, e a interação com os objetos se dá por *point-and-click* [55]. Entretanto, uma das limitações do procedimento é que a geração de regras contida no jogo é imutável, portanto impede a aplicação da DSL para outro domínio que não seja o definido (tratamento do transtorno do espectro autista) [56]. Além disso, a ferramenta foi idealizada apenas para uso, e necessita ainda de conhecimento técnico para a criação dos jogos. Ademais, a validação, apesar de contar com a presença dos especialistas citados anteriormente, foi baseada em cenários fictícios, sem aplicação real.

Thillainathan e Leimeister [57] apresentaram um *framework* para a criação de jogos, com uma DSL própria chamada de *Serious Game Logic and Structure Modeling Language* (GLiSMO) e o ambiente de programação visual chamado de *Visual Programming Environment for Serious Games* (VIPEr). Esse *framework* é direcionado para a criação de jogos sérios em plataformas móveis ou navegadores *Web*, e a criação é feita por meio de "*point-and-click*". Por meio de *constructs* que descrevem o problema, a DSL é usada tanto para criação da lógica como para a criação do comportamento do jogo, sendo uma "ponte" entre a *interface* (VIPEr) e o restante das ferramentas de processamento do MDGD. Por meio de um estudo de caso chamado *Shack City*, um jogo 2D interativo dedicado a aprendizes na área sanitária, apresenta-se uma cidade virtual como pano de fundo na qual o jogador deve solucionar problemas fornecidos por personagens. Porém, destaca-se, como limitação, que a capacidade da DSL está limitada apenas à configuração de comportamentos e valores dentro do jogo, sendo o processo de construção do ambiente virtual automatizado e prefixado.

De Troyer et al. [58] apresentaram uma DSL com o principal objetivo de edição de elementos narrativos e de aspectos pedagógicos de JS baseado em narrativa, para especialistas sem conhecimento técnico no desenvolvimento de jogos. O uso de uma linguagem natural, como a língua inglesa, permite a escrita natural do conteúdo do jogo, por meio de tijolos que darão a sequência de início ao fim. As anotações fazem parte do aprendizado proposto ou controle da estrutura do jogo, e distingue-se da estrutura usada para a descrição da narrativa. A presença de um simulador para a validação semântica do modelo auxilia o processo de desenvolvimento para encontrar erros mais rapidamente. Portanto, essas características favorecem a criação de jogos em formato de texto de aventura. A principal limitação é a pouca capacidade de integração, dado a complexidade de fazer a comunicação do conteúdo pedagógico com os componentes visuais ou que estão relacionados à jogabilidade central do jogo.

D. Criação de jogos do gênero quiz com MDGD

Dois trabalhos se assemelham a este artigo com relação a jogabilidade do gênero *quiz* [59], [60]. Garcia et al. [59] propuseram uma DSL para a criação de jogos 2D do tipo *quiz* em contextos educacionais, para serem executados em qualquer navegador *Web* por meio do *HTML5*. Os *quizzes* criados oferecem também painéis para a leitura do material didático pelo jogador, que podem ser apresentados antes da resolução de perguntas do conteúdo abordado. No final do questionário ofertado, é apresentada uma pontuação total para o jogador. A possibilidade da questão ter múltiplas respostas e do professor escrever suas próprias questões estão com a DSL entre as principais inovações para o gênero desse jogo. Além disso, o trabalho descreveu um formato único para sua avaliação, ao testar a qualidade dos jogos produzidos e se a DSL cumpria o objetivo de facilitar a produção de jogos para pessoas sem conhecimento técnico em programação, dado o tempo necessário de produção. Com a análise dos dados quantitativos obtidos, houve a confirmação que a ferramenta facilitou a criação de *quizzes*.

Nyameiono et al. [60], por outro lado, acrescentaram elementos de jogos para um *software* educacional, para melhorar a familiaridade e conscientização de diretrizes de práticas clínicas. Essa experiência gamificada é feita por uma geração de perguntas do conteúdo de forma automática. Diferentemente dos outros trabalhos, nesse caso, a modelagem do problema (seleção de perguntas, perfil do usuário e objetos na tela) se baseia na pesquisa do *framework* do diagrama de predicados (DPF, do inglês *Diagram Predicate Framework*), que em vez de se basear em UML para construir o metamodelo, faz uso de uma modelagem formal por meio da teoria de categorias e transformações de grafos, que foi usada para garantir a corretude dos modelos [61]. A avaliação foi realizada por meio de um percurso cognitivo, no qual os especialistas da área relatam as respostas selecionadas em cenários com maior dificuldade. Embora os resultados desses experimentos mostram a utilidade dos jogos criados, não fica claro o melhor custo benefício - com relação a um gasto maior de tempo para garantir a corretude - da abordagem formal

usada se comparado a outros trabalhos que fizeram uso da EMF (*Eclipse Modelling Framework*) [62].

E. Discussões sobre os trabalhos relacionados

Embora as metodologias apresentadas para a produção de jogos sérios abordam os aspectos de jogo e de *design*, artefatos como a ferramenta de edição e a DSL presentes no MDGD não são contempladas no ciclo de desenvolvimento. Em particular, os requisitos da DSL devem ser analisados mais cuidadosamente em conjunto com o tema, pois alterações implicam na corretude dos modelos. Tais mudanças também impactam na ferramenta de edição, já que está dependente dos requisitos obtidos para a DSL e do metamodelo a ser construído.

Van Deursen Klint e Visse [46] providenciam passos gerais para serem aplicados no desenvolvimento no contexto da engenharia de *software*, mas não é feito para o desenvolvimento de jogos. Portanto, há uma carência com relação à abordagem do *design* do jogo e também da inserção de recursos pedagógicos. Além disso, não cobre o uso de outras ferramentas de integração, como motores de jogos, para fazer o encadeamento da transformação de artefatos. Também assume a necessidade do desenvolvimento do compilador próprio, quando há ferramentas que diminuem consideravelmente as traduções sintáticas e semânticas envolvidas.

De forma geral, os trabalhos analisados no gênero *quiz* e aventura buscaram mapear os conteúdos que estão presentes pela edição da DSL, mas não o fizeram por meio de uma metodologia que realize a integração de diferentes aspectos, como: pessoas, artefatos, procedimentos e suas atividades. Assim, existe uma ausência nos requisitos definidos para construção da ferramenta de edição e da DSL. Embora especialistas tenham avaliado a capacidade da DSL de um trabalho [59], os testes devem ser estendidos para a ferramenta de edição e durante todo o processo de desenvolvimento. A presença desses problemas, que ocorrem em variadas fases do desenvolvimento, sugerem a necessidade de um processo para sistematização de um MDGD para jogos sérios, e é apresentado na próxima seção.

IV. Processo para Criação e Desenvolvimento Orientado a Modelos de Jogos Sérios

O objetivo do processo de apoio à criação e desenvolvimento orientado a modelos de jogos sérios é propor uma visão geral e de alta abstração, com a especificação de recursos e procedimentos comuns, considerando ser estendido para qualquer gênero de jogabilidade, que guiem a análise, o *design*, a implementação e o uso de MDGD para jogos sérios. Antes de apresentar as fases com suas atividades definidas, que servirão de guia para a construção da DSL, foi necessário definir: (i) os perfis e papéis das pessoas envolvidas e (ii) artefatos produzidos e sua função.

Os papéis presentes no processo do MDGD são: (a) **Especialista**, responsável por criar o conteúdo e introduzir os parâmetros relacionados ao jogo por meio da ferramenta de edição; (b) **Programador**, responsável por definir e criar as regras de transformação por meio do dicionário; (c) ***Designer***, responsável por listar e criar os elementos de jogabilidade e gráficos necessários no jogo; e (d) **Aprendiz**, usuário final, ou seja, o jogador alvo envolvido com o objetivo do JS.

Com relação aos artefatos produzidos, encontram-se dois: a **DSL** e a ferramenta gráfica de edição ou simplesmente **ferramenta de edição**. A DSL é a linguagem de programação que será usada para a produção do jogo sério de um gênero específico e possui sua expressividade definida pelo dicionário e, consequentemente, seu metamodelo. Para que essa linguagem seja usada por especialistas, a ferramenta de edição facilitará o manuseio da linguagem ao eliminar conhecimento técnico de programação.

O processo para a criação de DSL para jogos sérios apresentado neste artigo é baseado nas fases e atividades propostas por Van Deursen et al. [46] com adaptações necessárias dadas as particularidades do desenvolvimento de jogos sérios. Essas fases fornecem uma sequência sistematizada de passos que podem ser usados na elaboração de uma DSL para qualquer domínio. Porém, esse procedimento é específico para o desenvolvimento de *softwares* com objetivos definidos. Portanto, são necessárias adaptações, auxiliadas pela literatura existente, que reflitam o processo de desenvolvimento de jogos sérios. Na fase de análise, a partir dos requisitos apresentados em Rocha et al. [50], expande-se para a identificação dos objetivos do JS, os requisitos pedagógicos envolvidos e a definição do gênero e ambiente virtual. Além disso, fundamenta-se na delimitação do escopo proposta por Aslan e Balci [49] como forma de selecionar os elementos de jogabilidade e pedagógicos desejados. A escolha desses trabalhos é baseada na necessidade de informações para a idealização do jogo sério (tipo de jogabilidade e objetivos pedagógicos) e de impor restrições ao domínio que afetará a DSL. Dessa forma, as subseções a seguir apresentam as atividades de [46] (agrupadas nas fases de análise, *design*, implementação e uso) adaptadas para a criação de jogos sérios, a partir nas metodologias de desenvolvimento de jogos sérios descritas por [50] e [49]. A **Fig.** 1 representa a modelagem de todo o processo no padrão *Business Process Modeling Notation* (BPMN) [63], com gerenciamento das tarefas para cada uma das fases. Nas próximas subseções, serão apresentados como os artefatos e papéis estão envolvidos em cada fase.

A. Análise

A fase de **(1) Análise**, baseada na fase de mesmo nome proposta por [46], envolve os perfis do especialista e da equipe de desenvolvimento (*designer* e programador) e compreende as atividades de **(1.1) Identificar o domínio do problema e seus requisitos** e **(1.2) Coletar informações relevantes sobre o domínio**, tanto para a DSL como para a ferramenta de edição.

No caso da atividade **(1.1)**, para a DSL, é necessário estabelecer: o problema que está diretamente ligado com o objetivo do jogo sério, os requisitos pedagógicos necessários (o efeito desejado (o que) e de qual forma se dará para alcançá-lo (como) [50]), o ambiente virtual (3D ou 2D) e o gênero (jogabilidade) do jogo sério. Para a ferramenta de edição,

deve ser considerado: gerar ou impedir a geração do jogo caso o modelo apresente algum problema; enviar mensagens de conserto e sugestões que sigam uma linguagem comum e acessível para qualquer perfil; e identificar requisitos de funções básicas (criar, modificar, selecionar e excluir) para elementos pedagógicos e de jogabilidade, sincronizando com as informações já coletadas para a DSL.

No passo **(1.2)**, as informações para a DSL incluem: objetivos pedagógicos, interface gráfica e mecânicas do gênero específico [49]. A finalidade dessa atividade é delimitar o escopo que impacta diretamente no domínio e funcionalidades da DSL. No caso da ferramenta de edição, identificam-se quais funcionalidades devem ser descartadas ou quais que serão implementadas para facilitar a manipulação gráfica e a criação estrutural do jogo sério.

Antes de prosseguir para o *design*, é feita uma confirmação dos requisitos e informações levantadas com, pelo menos, os *designers* e especialistas. Opcionalmente, por se tratar da validação em alto nível, essa confirmação pode incluir outros atores da equipe e o próprio aprendiz, a depender da aplicação.

B. Design

As regras de transformação, que são elaboradas na fase **(2) *Design***, servem como base para a expressão de elementos e suas relações na DSL. Pela necessidade de atividades do *design* do JS, essa é uma fase acrescentada quando comparada ao processo de [46]. Foi adicionada a atividade de "Projeto do jogo" pois é necessário verificar se o mapeamento se aproxima, de fato, da ideia geral do jogo. O *Design* compreende as atividades: **(2.1) Mapear as informações** em um conjunto semântico de notações, operações e restrições; **(2.2) Projetar DSL** que descreva concisamente as aplicações no domínio, e **(2.3) Projetar jogo sério**. Todas as atividades envolvem o *designer* (mais responsabilidades) e o programador.

No passo **(2.1)**, as informações coletadas servem como base dos elementos que compõem o JS para o mapeamento das notações (nome dos elementos, tais como, propriedades e tipos) e suas operações (como relações de dependências e associações). Também inclui o mapeamento de restrições para evitar comportamentos de programação indesejados na DSL.

Feito o mapeamento, a atividade **(2.2)** consiste no projeto do dicionário que representará as regras de transformação da DSL, sendo baseada no mapeamento das informações, e a escrita no *framework* escolhido. Deve garantir a expressividade necessária para construção de mais de um jogo.

No passo **(2.3)**, é feito o projeto do jogo, para garantir que o jogo possa, de fato, ser construído com elementos e relações previstos na ferramenta de edição. A visão do mapeamento de informações deve estar de acordo com a visão de projeto do jogo, para que o programador possa implementá-lo.

Antes da *implementação*, é feita uma confirmação dos elementos e suas relações e do protótipo *lo-fi* com, pelo menos, os programadores e especialistas envolvidos. Preferencialmente, pela importância da validação dos primeiros artefatos concretos, em alto nível, que representam a DSL e jogos sérios visados, essa confirmação pode incluir outros atores da equipe e o próprio aprendiz.

C. Implementação

A fase **(3) Implementação** é baseada na fase de mesmo nome proposta por [46] e estendida com a atividade de implementação da ferramenta de edição. Compreende as atividades de: **(3.1) Construir uma ferramenta de transformação** que implemente as notações semânticas; **(3.2) Projetar e implementar um analisador sintático**; **(3.3) Projetar e implementar um verificador semântico**; e **(3.4) Implementar a ferramenta de edição**.

No passo **(3.1)**, a construção das ferramentas de transformação com as notações semânticas são implementadas pelo programador com o dicionário e sua sintaxe concreta (*framework* escolhido na atividade 2 na fase de "Design". No caso de **(3.2)**, o *parser* é implementado pelo programador, ou gerado por geradores de *parsers*, garantindo a corretude da sintaxe da linguagem. Em seguida, em **(3.3)**, o verificador é implementado pelo programador; ou as verificações são implementadas no código-fonte gerado específico para esse fim, capturando erros que o analisador sintático não é capaz de encontrar. Na atividade **(3.4)**, é feita a implementação da ferramenta de edição, pelo programador, com base nos requisitos coletados na fase de "Análise", compreendendo funcionalidades e o mapeamento dos elementos e suas relações para o *design* dos jogos.

Antes da fase de *uso*, é feita a confirmação da ferramenta de transformação e da DSL implementada, por meio de versões parciais das ferramentas citadas e protótipos de testes.

D. Uso

A fase **(4) Uso**, baseada na fase de mesmo nome proposta por [46], compreende as atividades: **(4.1) Escrever os jogos sérios com a DSL** para as aplicações desejadas. O especialista de domínio, por meio da ferramenta de edição, realiza a edição dos elementos de conteúdo e de jogabilidade, construindo o jogo final para os aprendizes; e **(4.2) Jogar o jogo sério.** O aprendiz usa o executável do JS criado.

V. Linguagem de Modelagem para Jogos de Aventuras e Quiz

A DSL nomeada LMJAQ (Linguagem de Modelagem para Jogos de Aventura e Quiz) é proposta para dar maior criatividade e engajamento para a criação de jogos sérios do tipo *quiz* e aventura, baseada no processo de desenvolvimento apresentado na Seção IV. A linguagem visa preencher lacunas que envolvem a capacidade de edição de elementos narrativos e um maior desafio, além da necessidade de integrar as diferentes perspectivas (pedagógica, jogabilidade e narrativa).

A. Ferramental

O *Eclipse Modelling Framework* (EMF) é a principal ferramenta usada para edição da linguagem para a criação da DSL [62] e geração de código. A escolha foi baseada na evolução da ferramenta e do alto volume de documentação

existente para o manuseio, e a geração de modelos no formato ".xmi".

São duas linguagens de programação usadas na escrita: *Xtext* e *Xtend*. A primeira é um *framework* usado para desenvolver linguagens de programação ou DSL(s), ou seja, para gerar o *parser* e a AST. A segunda é usada como alternativa ao Java, como a linguagem de programação que descreve o código fonte dos *parsers*, validadores e testes. Dado que é sintaticamente e semanticamente em Java, recomenda-se o uso dessa linguagem para diminuir a verbosidade, e por conta da presença do conceito de métodos de extensões e expressões *lambda* que aumentam a quantidade de formas existentes para estender bibliotecas.

As próximas subseções se dedicam ao uso do processo descrito na seção IV para o caso específico da LMJAQ.

B. Análise

Com base na fase **(1) Análise** explorada na Seção IV-A, para a atividade **(1.1)**, foram identificados os problemas e requisitos da DSL e da ferramenta de edição. Para a DSL, a linguagem (LMJAQ) é direcionada a criação de jogos 2D de aventura na forma de um *quiz*. Espera-se que o aprendizado aconteça com as respostas das perguntas e seus efeitos. Os principais requisitos que a DSL deve contemplar, em relação ao JS criado, são: (a) efeito desejado: aprendizado do conteúdo envolvido em uma questão; (b) forma: efeito das ações do jogador na narrativa e *feedback* visual e textual; (c) ambiente virtual: 2D; e (d) gênero: *quiz* e aventura.

Para a ferramenta de edição, deve-se compreender tarefas básicas para modificar elementos básicos, distinção visual (cores, formatos) entre elementos com diferentes finalidades (*quiz*, pedagógico, jogabilidade e aventura), ter funcionalidades intuitivas e providenciar dicas na representação como ao construir suas relações (uma questão deve conter alternativas, por exemplo). Os principais requisitos são: (a) criar, copiar, selecionar, modificar e excluir elementos; (b) arrastar e soltar elementos para facilitar a função de mover; (c) distinguir visualmente os elementos pertencentes aos gêneros (*quiz* e aventura) e pedagógicos; e (d) providenciar dicas enquanto o ponteiro do *mouse* estiver em cima de elementos e como acrescentar suas relações (*tooltip*).

Na atividade **(1.2)**, foi feita a coleta de informações relevantes que descreverão os elementos e suas relações na LMJAQ. A começar pela DSL, essa descrição é verificável pelos requisitos (pedagógicos e de jogabilidades, do jogo *quiz* e do jogo aventura) aos JS que poderão ser criados.

Requisitos Pedagógicos e de Jogabilidade: (a) alterar o conteúdo do jogo (personagens, itens, efeitos da resposta do jogador, textos); (b) alterar o conteúdo audiovisual (cenários, imagens e sons); (c) definir estados do progresso de jogo (sendo exigido, no mínimo, inicial e final); (d) incluir diferentes formas de *feedback* (visual, sonoro e textual); (e) incluir um sistema de pontuação e sua apresentação para o jogador. (f) incluir objetos pré-definidos (questão, alternativa, estado, imagem, efeito, personagem) baseado nas funções propostas do jogo para facilitar o *design* de telas; (g) associar consequências da ação do jogador com objetivos pedagógicos; e (h) incluir a possibilidade de exibir material didático (textual e imagem).

Requisitos do jogo *Quiz*: (a) criar uma sequência de perguntas; (b) apresentar uma pergunta e suas respectivas opções de respostas; (c) permitir a inclusão de dificuldade por questão ou do conjunto; (d) permitir tipos diferentes de pergunta (múltipla escolha, verdadeiro ou falso); (e) garantir que as respostas para uma pergunta sigam um mesmo formato: imagens, texto ou áudio; e (f) editar propriedades da pergunta e do *quiz* (dificuldade, tempo, quantidade de respostas, possibilidade de uso do item, tempo para resposta).

Requisitos do jogo *Aventura*: (a) editar atores: tipo (NPC ou jogador), papel (auxiliar, divisor), identidade (nome); (b) editar recursos disponíveis: consumação (consumível, permanente), efeitos; e (c) editar ambiente: história do jogo, gênero complementares (horror, comédia, ação, etc.), estado inicial, final e ramificações.

Para a ferramenta de edição, foram levantados os objetos predefinidos que são baseados nos recursos pedagógicos e de jogabilidade. O resultado inicial é apresentado na **Tabela** I, com os itens que serão objetos predefinidos. Esses itens foram selecionados a partir dos trabalhos [64] (quiz), [8] (aventura), [50] (recursos pedagógicos) e [49] (jogabilidade em geral).

C. Design

Baseado na fase **(2) *Design*** apresentada em IV-B, apresentam-se os objetos criados da atividade "Mapear essas informações em um conjunto semântico de notações e operações": (a) **DataType**: representa os objetos como dados elementares com inteiros e strings (*Eint* e *EStr* no EMF). (b) **GameType**: representa os tipos de jogos indicados, dividindo o objeto em Aventura ou *quiz*. Cada gênero conterá os elementos e atributos necessários para dar a capacidade de edição necessária para a jogabilidade. Os elementos do gênero *quiz* são baseados no trabalho apresentado em [64] e, no caso de aventura, é baseado nos itens apresentados em [8]. (c) **Pedagogical**: representa o objeto onde contém toda a capacidade de adicionar elementos pedagógicos oferecidos pela DSL. (d) **Audiovisual**: representa o objeto onde contém toda a capacidade de adicionar elementos multimídia (imagens e sons) oferecidos pela DSL.

VI. Discussão

Discutem-se, nesta seção, os principais resultados, contribuições e desafios encontrados na elaboração do processo, e na DSL proposta baseado nele. São quatro aspectos analisados: (1) método e processos, (2) projeto, (3) pessoas e (4) artefatos criados e produto final (conforme em Aslan e Balci [49]).

No caso do método, o uso das fases e atividades propostas por [46] serviram como base para a criação de um processo visado ao desenvolvimento de jogos sérios. Essa base integrada com a jogabilidade e os recursos pedagógicos nas abordagens dedicadas a jogos sérios permitiu, como contribuição, a concepção de um processo organizado e sistematizado para

TABELA I
OBJETOS PREDEFINIDOS A PARTIR DE COLETA DE INFORMAÇÃO PARA A DSL E FERRAMENTA DE EDIÇÃO.

<table>
<tr><th>Recursos Pedagógicos</th><th colspan="2">Recursos de Jogabilidade</th></tr>
<tr><td rowspan="2">Janela com instruções (texto) e feedback (textual, sonoro, visual)</td><td colspan="2">Formato de apresentação (texto, imagem e som), estados (entradas, saídas e condições) e feedback (formato)</td></tr>
<tr><td>Quiz: questões (dificuldade, texto da pergunta, pontuação, formato, tipo, tempo para responder), alternativas (formato, efeito, transição de estado)</td><td>Aventura: estados (constantes: início e fim), itens, inventário, ambiente (tema, gênero e complementar)</td></tr>
</table>

a produção de DSLs para jogos sérios. Entretanto, o processo proposto só foi usado para Análise e Mapeamento das informações da DSL e ferramenta de edição, e ainda deverá ser usado, em trabalhos futuros, para Projeto, Implementação, Uso e Avaliação da DSL LMJAQ. Além disso, dado que o processo visa ser usado para criação de diversos gêneros de JS, ele ainda deve ser usado em outros contextos e poderá ser refinado para uma maior abrangência, principalmente com inclusões de atividades de teste e avaliação na fase de "Uso'.

Em relação às pessoas, o processo visou incluir todos os papéis existentes em cada atividade relacionada com a função. Em trabalhos futuros, é possível aperfeiçoar o diagrama BPMN para que seja mais detalhado na relação tarefa com o papel (pois algumas atividades são realizadas por dois profissionais em conjunto). Uma primeira avaliação com especialistas será importante, como trabalhos futuros, para encontrar melhorias no projeto.

Em relação aos artefatos criados, destaca-se, como contribuição, a construção e esquematização do processo no formato de BPMN, como documento visual para organizar as tarefas necessárias. Neste trabalho, a DSL LMJAQ foi contemplada apenas nas fases de análise e *design* (com criação dos documentos da DSL, ferramenta de Edição e Conjunto Semântico), seguidas de uma avaliação preliminar; os trabalhos futuros incluem a implementação dos artefatos (a ferramenta de edição, a ferramenta de transformação e seus componentes e a DSL), além de um jogo sério, por meio da linguagem desenvolvida, do gênero *quiz* e aventura.

VII. CONCLUSÃO

Esse trabalho propôs um novo processo para o desenvolvimento baseado em modelos de DSLs para jogos sérios, com o objetivo de beneficiar maior criatividade e adaptabilidade para os diferentes gêneros de jogabilidade e temas. Foi realizada uma avaliação parcial desse processo por meio da (1) aplicação da fase "Análise" e "*Design*", para a LMJAQ para a criação de jogos sérios de gêneros *quiz* e aventura, e da (2) discussão sobre o método, pessoas, artefatos e projeto.

Os trabalhos futuros envolvem o *Design* da ferramenta de edição, bem como a implementação de todos os artefatos (ferramenta de edição e DSL) que estão compreendido na fase de "Implementação". Espera-se uma validação após concretização para buscar opiniões de especialistas de diferentes perfis para aprimoramento.

AGRADECIMENTOS

Os autores agradecem o financiamento concedido pela CAPES.

REFERÊNCIAS

[1] T. Susi, M. Johannesson, and P. Backlund, "Serious games: An overview," 2007.

[2] A. De Gloria, F. Bellotti, and R. Berta, "Serious games for education and training," *International Journal of Serious Games*, vol. 1, no. 1, 2014.

[3] P. Wouters, E. D. Van der Spek, and H. Van Oostendorp, "Current practices in serious game research: A review from a learning outcomes perspective," *Games-based learning advancements for multi-sensory human computer interfaces: techniques and effective practices*, pp. 232–250, 2009.

[4] J. Fox, L. Pittaway, and I. Uzuegbunam, "Simulations in entrepreneurship education: Serious games and learning through play," *Entrepreneurship Education and Pedagogy*, vol. 1, no. 1, pp. 61–89, 2018.

[5] V. Wattanasoontorn, I. Boada, R. García, and M. Sbert, "Serious games for health," *Entertainment Computing*, vol. 4, no. 4, pp. 231–247, 2013.

[6] A. Azadegan, J. C. Riedel, and J. B. Hauge, "Serious games adoption in corporate training," in *International conference on serious games development and applications*. Springer, 2012, pp. 74–85.

[7] B. Bontchev and D. Vassileva, "Educational quiz board games for adaptive e-learning," in *Proc. of Int. Conf. ICTE*, 2010, pp. 63–70.

[8] E. Ju and C. Wagner, "Personal computer adventure games: their structure, principles, and applicability for training," *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, vol. 28, no. 2, pp. 78–92, 1997.

[9] C. Harteveld, R. Guimarães, I. Mayer, and R. Bidarra, "Balancing pedagogy, game and reality components within a unique serious game for training levee inspection," in *International conference on technologies for e-learning and digital entertainment*. Springer, 2007, pp. 128–139.

[10] P. Rooney, "A theoretical framework for serious game design: exploring pedagogy, play and fidelity and their implications for the design process," *International Journal of Game-Based Learning (IJGBL)*, vol. 2, no. 4, pp. 41–60, 2012.

[11] F. Mehm, "Authoring serious games," in *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, 2010, pp. 271–273.

[12] A. Yessad, J.-M. Labat, and F. Kermorvant, "Segae: A serious game authoring environment," in *2010 10th IEEE International Conference on Advanced Learning Technologies*. IEEE, 2010, pp. 538–540.

[13] D. Charsky, "From edutainment to serious games: A change in the use of game characteristics," *Games and culture*, vol. 5, no. 2, pp. 177–198, 2010.

[14] S. Theodosiou and I. Karasavvidis, "Serious games design: A mapping of the problems novice game designers experience in designing games," *Journal of e-Learning and Knowledge Society*, vol. 11, no. 3, 2015.

[15] A. Slootmaker, H. Hummel, and R. Koper, "Evaluating the usability of authoring environments for serious games," *Simulation & gaming*, vol. 48, no. 4, pp. 553–578, 2017.

[16] T. Murray, "Design tradeoffs in usability and power for advanced educational software authoring tools," *Educational Technology*, vol. 44, no. 5, pp. 10–16, 2004.

[17] S. Arnab, T. Lim, M. B. Carvalho, F. Bellotti, S. De Freitas, S. Louchart, N. Suttie, R. Berta, and A. De Gloria, "Mapping learning and game mechanics for serious games analysis," *British Journal of Educational Technology*, vol. 46, no. 2, pp. 391–411, 2015.

[18] M. Zhu and A. I. Wang, "Model-driven game development: A literature review," *ACM Comput. Surv.*, vol. 52, no. 6, Nov. 2019. [Online]. Available: https://doi.org/10.1145/3365000

[19] S. Tang, M. Hanneghan, and C. Carter, "A platform independent game technology model for model driven serious games development," *Electronic Journal of e-Learning*, vol. 11, no. 1, pp. 61–79, 2013.

[20] F. Hamiye, B. Said, and B. Serhan, "A framework for the development of serious games for assessment," in *Games and Learning Alliance*, A. Liapis, G. N. Yannakakis, M. Gentile, and M. Ninaus, Eds. Cham: Springer International Publishing, 2019, pp. 407–416.

[21] S. Tang and M. Hanneghan, "Game content model: an ontology for documenting serious game design," in *2011 Developments in E-systems Engineering*. IEEE, 2011, pp. 431–436.

[22] H. Cardona-Reyes, J. Muñoz-Arteaga, L. Barba-González, and G. Ortiz-Aguiñaga, "Model-driven multidisciplinary production of virtual reality environments for elementary school with adhd," in *Iberoamerican Workshop on Human-Computer Interaction*. Springer, 2020, pp. 181–192.

[23] T. H. Apperley, "Genre and game studies: Toward a critical approach to video game genres," *Simulation & Gaming*, vol. 37, no. 1, pp. 6–23, 2006.

[24] D. Arsenault, "Video game genre, evolution and innovation," *Eludamos. Journal for computer game culture*, vol. 3, no. 2, pp. 149–176, 2009.

[25] D. Guimarães, "Kahoot: quizzes, debates e sondagens," *Apps para dispositivos móveis: manual para professores, formadores e bibliotecários*, pp. 203–224, 2015.

[26] B. Cavallari, J. Heldberg, and B. Harper, "Adventure games in education: A review," *Australasian journal of educational technology*, vol. 8, no. 2, 1992.

[27] E. Vockell and P. LaRear, *The computer in the foreign language curriculum*. Alfred A. Knopf, Inc., 1988.

[28] W. Shi, K. Kaneko, C. Ma, and Y. Okada, "A framework for automatically generating quiz-type serious games based on linked data," *International Journal of Information and Education Technology*, vol. 9, no. 4, pp. 250–256, 2019.

[29] R. G. Mangowal, U. L. Yuhana, E. M. Yuniarno, and M. H. Purnomo, "Mathbharata: A serious game for motivating disabled students to study mathematics," in *2017 IEEE 5th International Conference on Serious Games and Applications for Health (SeGAH)*. IEEE, 2017, pp. 1–6.

[30] P. Meera, M. McLain, K. Bijlani, R. Jayakrishnan, and B. R. Rao, "Serious game on flood risk management," in *Emerging research in computing, information, communication and applications*. Springer, 2016, pp. 197–206.

[31] D. R. Michael and S. L. Chen, *Serious games: Games that educate, train, and inform*. Muska & Lipman/Premier-Trade, 2005.

[32] J. P. Gee, "Learning by design: Good video games as learning machines," *E-learning and Digital Media*, vol. 2, no. 1, pp. 5–16, 2005.

[33] M. Prensky, "Digital game-based learning," *Computers in Entertainment (CIE)*, vol. 1, no. 1, pp. 21–21, 2003.

[34] R. V. Rocha, I. I. Bittencourt, and S. Isotani, "Análise, projeto, desenvolvimento e avaliação de jogos sérios e afins: uma revisão de desafios e oportunidades," in *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, vol. 26, no. 1, 2015, p. 692.

[35] J. Novak, "Desenvolvimento de games," *São Paulo: Cengage Learning*, pp. 354–355, 2010.

[36] A. Pho and A. Dinscore, "Game-based learning," *Tips and trends*, 2015.

[37] I. F. Silveira, "Open educational games: Challenges and perspectives," in *2016 XI Latin American Conference on Learning Objects and Technology (LACLO)*. IEEE, 2016, pp. 1–9.

[38] M. Oliveira, J. Crowcroft, and M. Slater, "An innovative design approach to build virtual environment systems," in *Proceedings of the workshop on Virtual environments 2003*, 2003, pp. 143–151.

[39] S. Tang and M. Hanneghan, "State of the art model driven game development: A survey of technological solutions for game-based learning," *Journal of Interactive Learning Research*, vol. 22, pp. 551–605, 01 2011.

[40] M. Fowler, "Language workbenches: The killer-app for domain specific languages," 2005.

[41] M. Voelter, S. Benz, C. Dietrich, B. Engelmann, M. Helander, L. C. Kats, E. Visser, G. Wachsmuth *et al.*, *DSL engineering: Designing, implementing and using domain-specific languages*. dslbook. org, 2013.

[42] D. Thomas and B. M. Barry, "Model driven development: the case for domain oriented programming," in *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, 2003, pp. 2–7.

[43] M. Fowler, *Domain-specific languages*. Pearson Education, 2010.

[44] A. V. Aho, R. Sethi, and J. D. Ullman, "Compilers, principles, techniques," *Addison wesley*, vol. 7, no. 8, p. 9, 1986.

[45] M. A. Dave, "Compiler verification: a bibliography," *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 6, pp. 2–2, 2003.

[46] A. Van Deursen, P. Klint, and J. Visser, "Domain-specific languages: An annotated bibliography," *ACM Sigplan Notices*, vol. 35, no. 6, pp. 26–36, 2000.

[47] A. Meduna, *Formal languages and computation: models and their applications*. CRC Press, 2014.

[48] R. Wilhelm, H. Seidl, and S. Hack, *Compiler design: syntactic and semantic analysis*. Springer Science & Business Media, 2013.

[49] S. Aslan and O. Balci, "Gamed: digital educational game development methodology," *Simulation*, vol. 91, no. 4, pp. 307–319, 2015.

[50] R. V. Rocha, P. H. Valle, J. C. Maldonado, I. I. Bittencourt, and S. Isotani, "Aimed: agile, integrative and open method for open educational resources development," in *2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2017, pp. 163–167.

[51] B. Cowan and B. Kapralos, "A survey of frameworks and game engines for serious game development," in *2014 IEEE 14th International Conference on Advanced Learning Technologies*. IEEE, 2014, pp. 662–664.

[52] A. E. Olszewski and T. A. Wolbrink, "Serious gaming in medical education: a proposed structured framework for game development," *Simulation in Healthcare*, vol. 12, no. 4, pp. 240–253, 2017.

[53] N. Iuppa and T. Borst, *End-to-end game development: creating independent serious games and simulations from start to finish*. CRC Press, 2012.

[54] P. Laforcade and Y. Laghouaouta, "Generation of Adapted Learning Game Scenarios: A Model-Driven Engineering Approach," in *Computer Supported Education*, ser. Comm. in Computer and Information Science. Springer, Jun. 2019, vol. 1022, pp. 95–116. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02177667

[55] E. Corkill, "Real escape game brings its creator's wonderment to life," *Pozyskano z http://www. japantimes. co. jp/life/2009/12/20/to-be-sorted/real-escapegame-brings-its-creators-wonderment-to-life/#. V-BG6vCLShd*, 2016.

[56] M. C. T. V. Teixeira, T. P. Mecca, R. d. L. Velloso, R. B. Bravo, S. H. B. Ribeiro, M. T. Mercadante, and C. S. d. Paula, "Literatura científica brasileira sobre transtornos do espectro autista," *Revista da Associação Médica Brasileira*, vol. 56, no. 5, pp. 607–614, 2010.

[57] N. Thillainathan and J. M. Leimeister, "Educators as game developers—model-driven visual programming of serious games," in *Knowledge, Information and Creativity Support Systems*, S. Kunifuji, G. A. Papadopoulos, A. M. Skulimowski, and J. Kacprzyk , Eds. Cham: Springer International Publishing, 2016, pp. 335–349.

[58] O. De Troyer, F. Van Broeckhoven, and J. Vlieghe, *Creating Story-Based Serious Games Using a Controlled Natural Language Domain Specific Modeling Language*. Cham: Springer International Publishing, 2017, pp. 567–603. [Online]. Available: https://doi.org/10.1007/978-3-319-51645-5_25

[59] C. G. García, E. R. Núñez-Valdez, P. Moreno-Ger, R. G. Crespo, B. C. P. G-Bustelo, and J. M. C. Lovelle, "Agile development of multiplatform educational video games using a domain-specific language," *Universal Access in the Information Society*, vol. 18, no. 3, pp. 599–614, 2019.

[60] J. N. Nyameino, B.-R. Ebbesvik, F. Rabbi, M. C. Were, and Y. Lamo, "Model-driven automatic question generation for a gamified clinical guideline training system," in *Evaluation of Novel Approaches to Software Engineering*, E. Damiani, G. Spanoudakis, and L. A. Maciaszek, Eds. Cham: Springer International Publishing, 2020, pp. 227–245.

[61] A. Rutle, A. Rossini, Y. Lamo, and U. Wolter, "A diagrammatic formalisation of mof-based modelling languages," in *International Conference on Objects, Components, Models and Patterns*. Springer, 2009, pp. 37–56.

[62] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed., ser. Eclipse Series. Upper Saddle River, NJ: Addison-Wesley, 2009. [Online]. Available: https://www.safaribooksonline.com/library/view/emf-eclipse-modeling/9780321331885/

[63] OMG, *Business Process Model and Notation (BPMN), Version 2.0*, Object Management Group Std., Rev. 2.0, January 2011. [Online]. Available: http://www.omg.org/spec/BPMN/2.0

[64] R. Oliveira, G. D. Belarmino, D. M. Bouro, M. A. Oliveira, J. P. Motta, J. R. Pereira, E. S. Bezerra, R. M. Souza, C. L. Rodriguez, D. H. Goya *et al.*, "Game quiz: protótipo de uma plataforma para criação de jogos sérios do tipo quiz," 2020.

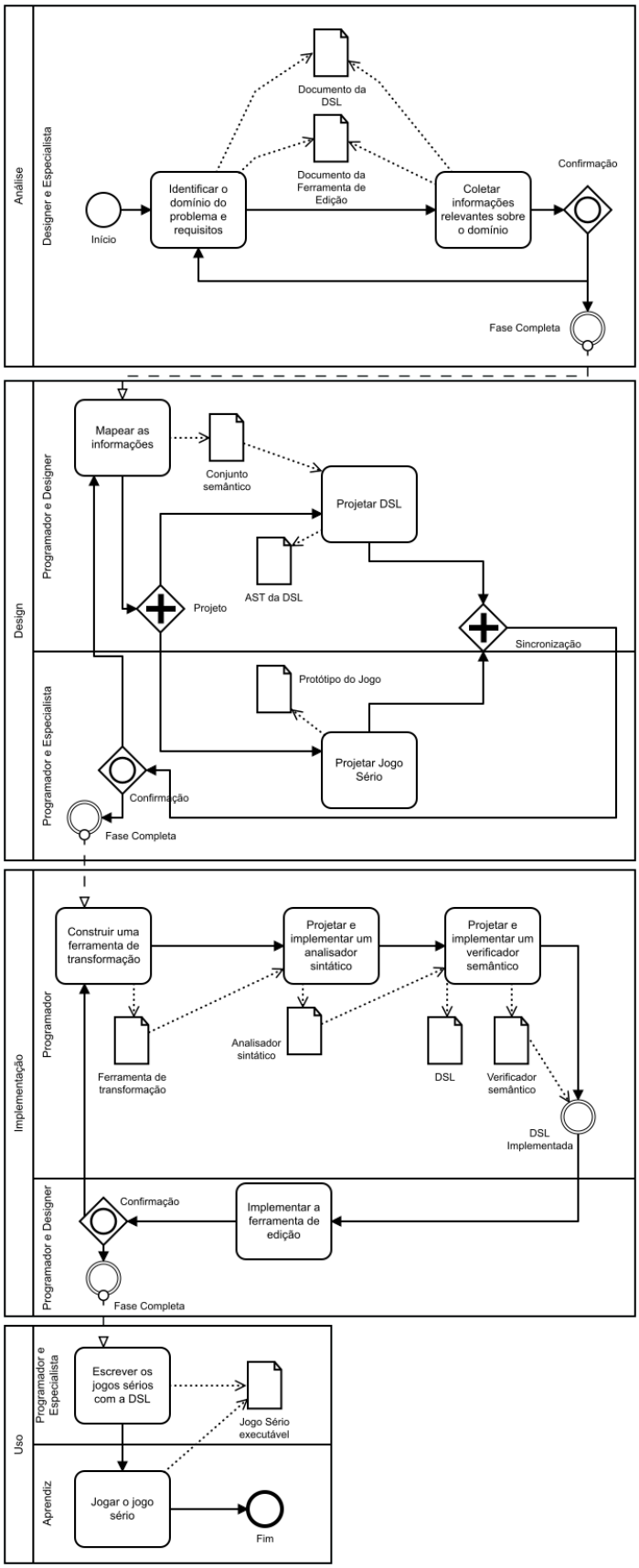

Fig. 1. Processo de apoio a criação e desenvolvimento orientado a modelos de jogos sérios, incluindo as tarefas necessárias e suas dependências (elaborado pelos autores).