Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Cenários Interativos para Padrões de Arquitetura de Software

Filipe José Haider

JUIZ DE FORA
FEVEREIRO, 2022

Cenários Interativos para Padrões de Arquitetura de Software

Filipe José Haider

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Sistemas de Informação

Orientador: Igor de Oliveira Knop

JUIZ DE FORA
FEVEREIRO, 2022

Cenários Interativos para Padrões de Arquitetura de Software

Filipe José Haider

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Igor de Oliveira Knop
D.Sc. Modelagem Computacional

André Luiz de Oliveira
D.Sc. Ciências da Computação e Matemática Computacional

Barbara de Melo de Quintela
D.Sc. Modelagem Computacional

JUIZ DE FORA
18 DE FEVEREIRO, 2022

Este trabalho é dedicado a Deus,

que sempre me ajudou nos momentos mais difíceis.

Resumo

O ensino de Engenharia de Software é em sua grande maioria ofertado aos alunos seguindo a forma expositiva tradicional. Dentre as alternativas para o ensino, os jogos sérios têm se mostrado uma ferramenta com bom potencial para a educação, pois são ambientes simulados que possibilitam testar, acertar e errar diversas vezes, respeitando o tempo cognitivo do educando, auxiliando para o seu aprendizado e aplicação prática. Mesmo assim, essa metodologia ainda não faz parte da cultura acadêmica, porém quando adotados, os professores optam por jogos sérios que já estejam prontos, visto a dificuldade e falta de conhecimento para a criação de novos softwares de ensino. Da mesma forma, narrativas interativas estão presentes na cultura como uma experiência na qual o leitor participa ativamente os resultados da história e são usadas na mídia impressa, audiovisual e como atividades em grupo. O uso de narrativas interativas tem grande potencial para auxiliar disciplinas teóricas que se utilizam de muitos estudos de casos. Este trabalho explora essa abordagem do uso de narrativas interativas na forma de um jogo sério para disciplinas de Engenharia de Software. As ferramentas mais populares para construção de narrativas interativas são avaliadas e duas histórias expõe do conteúdo de arquitetura de software, além da readaptação de uma história existente no contexto de desenvolvimento de software em Java. Por fim, um protótipo de um jogo sério é criado na forma de aplicativo para Android no qual os estudantes de engenharia de software podem ter acesso às histórias em seus dispositivos.

Palavras-chave: arquitetura de software; narrativas interativas; jogo sério.

Abstract

Software Engineering teaching usually is offered to students following the traditional expository form. Among the alternatives for teaching, serious games have proved to be a tool with good potential for education, as they are simulated environments that allow testing, correcting, and making mistakes several times, respecting the student's cognitive time, and practice. Therefore, this methodology is not yet part of the academic culture, but when adopted, teachers opt for serious games that are already ready, given the difficulty and lack of knowledge for creating new teaching software. Likewise, interactive narratives are present in culture as an experience in which the reader actively participates in the results of the story and are used in print, audiovisual, and group activities. The use of interactive narratives has great potential to help theoretical disciplines that use many case studies. This work explores this approach of using interactive narratives in the form of a serious game for Software Engineering disciplines. The most popular tools for building interactive narratives are evaluated and two stories expose the software architecture content, in addition to the re-adaptation of an existing story in the context of software development in Java. Finally, a prototype of a serious game is created in the form of an Android application in which software engineering students can access the stories on their devices.

Keywords: software architecture; interactive narrtives; serious games.

Agradecimentos

Dedico esse trabalho aos meus pais, meus maiores incentivadores, pela educação que me deram, pela disciplina que me ensinaram, pela dedicação nos cuidados, e por serem um verdadeiro pilar de esperança, sabedoria, respeito a Deus e amor em minha vida.

Ao professor Igor Knop por ter aceitado ser meu orientador e disponibilizar o seu tempo para me atender, seja nos horários combinados previamente ou até mesmo em situações de desespero no sábado a noite.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos ao longo desses vários anos de curso que de algum modo contribuíram para o meu enriquecimento pessoal e profissional.

"An unforeseen future nestled somewhere in time.
Unsuspecting victims no warnings no sign.
Judgement day the second coming arrives.
Before you see the light you must die".

Araya/Hanneman/King/Lombardo
(Slayer - South Of Heaven)

Conteúdo

Lista de Figuras

Lista de Abreviações e Siglas

CC Ciência da Computação. 12, 13

CSS *Cascading Style Sheets*. 21, 22

EC Engenharia de Computação. 13

ES Engenharia de Software. 9, 10, 12, 13, 23–25, 35

FIFO *(First In, First Out)*. 17

HTML *HyperText Markup Language*. 21, 22

HTTP HyperText Transfer Protocol. 15

Java EE Java Enterprise Edition. 23, 25, 26, 29, 32

JSP JavaServer Pages. 15

MBA Mestrado em Administração de Negócios. 22

MEC Ministério da Educação. 12

MVC *Model-View-Controller*. 7, 13–15, 27

SBC Sociedade Brasileira de Computação. 13

SCUMM *Script Creation Utility for Maniac Mansion*. 22

SI Sistemas de Informação. 13

TRB *Transportation Research Board*. 24

USC *Unified Soil Classification System*. 24

1 Introdução

1.1 Apresentação

Atualmente, a Engenharia de Software (ES) enfrenta desafio centrado na necessidade de criar mecanismos de ensino que proporcionem maior eficiência para assegurar o ensino-aprendizagem (SANTOS et al., 2008). Quando estão diante de um projeto satisfatório de metodologia de ensino, os estudantes se mostram desinteressados pela disciplina considerando-a teórica e burocrática, mostrando preferência pela escrita e funcionamento de programas a documentar formalmente os seus sistemas (SANTOS et al., 2008). Uma forma para melhorar a situação centra-se na adoção de técnicas de ensino alternativas como jogos e aplicativos (CHEN et al., 2008).

Nos últimos anos, a expectativa de que os jogos educacionais sejam uma alternativa bastante considerável para o ensino (DEUS, 2016). Tais jogos são desenvolvidos para serem aplicados em ambientes educacionais e como complemento às aulas de determinados assuntos. São criados observando aspectos educacionais e unem elementos inerentes a jogos, como as narrativas, regras, objetivos, competição e interação. Através de ambientes virtuais, ferramentas de simulação e determinados jogos, e as narrativas interativas para o ensino de engenharia de software (ANTUNES; PAIXãO; ALBUQUERQUE, 2019).

Os alunos adquirem capacidades, fruto de uma experiência com situações realísticas e "concretas" e que também são encontradas na prática. Segundo Amélio (2018) o mercado de jogos mais desenvolvido no mundo é o Norte Americano, que é referência para diversos países. No Brasil, nota-se fatores de influência desse consumo, pois os hardwares mais importantes no mercado local são importados ou trazidos dos Estados Unidos e China, com isso O mercado brasileiro apresentou crescimento nesse setor (ENRIQUEZ, 2021).

1.2 Descrição do Problema

As narrativas interativas para ensino de engenharia de software podem ser um método utilizado para auxiliar no ensino, visto que se observa uma escassez de mecanismos não convencionais para o ensino da arquitetura de software.

1.3 Justificativa

O presente trabalho justifica-se pela importância do engajamento dos alunos de engenharia de software, dado o crescente número de estudantes em todo mundo. Especificamente no Brasil onde há um grande investimento em educação ao nível superior e um baixo investimento na educação básica (CARVALHO, 2020).Observa-se também a falta de ferramentas e métodos alternativos, além de projeto-piloto que abordem, de maneira interativa, a ensino de engenharia de software.

1.4 Motivação

A principal motivação para este trabalho pauta-se na observação da utilização no uso de jogos sérios como um meio para resolver problemas relacionados ao desinteresse dos alunos pelos conteúdos da disciplina de ES. A criação de um aplicativo com a finalidade educacional, feito sob demanda para aulas, poderia diminuir a distância dos docentes no acesso a técnicas de aprendizado por meio dos jogos digitais, melhorando a transmissão de conteúdo em ES fazendo com que os alunos desfrutem mais intensamente do potencial dos jogos na educação.

1.5 Questões

Neste contexto, surgem as seguintes questões de pesquisa a serem respondidas com a realização deste trabalho:

- Quais modelos podem ser criados?
- Quais interações podem ser exploradas?

- Com o uso do dispositivo móvel, como é possível compartilhar e discutir a experiência de aprendizado entre alunos de uma determinada turma.

A partir dos questionamentos realizados, sob as quais os experimentos podem ser criados para validá-las ou refutá-las: (i) o conteúdo de arquitetura de software e sua dinâmica podem ser capturados por histórias interativas; se possível, (ii) as histórias interativas podem ser uma abordagem viável para um ensino; Ao aplicar a abordagem, (iii) a interatividade e das histórias, bem como a familiaridade das pessoas com os *smartphones*, aumenta o engajamento dos alunos com o assunto.

1.6 Objetivos

O objetivo geral é explorar uma metodologia ativa, através da criação de um protótipo digital, histórias interativas como um jogo sério para os conteúdos sobre padrões arquiteturais de software. Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

- Definir um modelo com os principais aspectos e dinâmicas de Arquitetura de Software;
- Compilar um conjunto de situações problema que reproduzem cenários e efeitos dessas ações do projeto;
- Interligar as situações em um aplicativo para dispositivos móveis de forma que os alunos possam interagir com a narrativa associada aos padrões arquiteturais;

1.7 Organização do Documento

Este trabalho está organizado em cinco capítulos. Além desta Introdução, no Capítulo 2 é apresentada a fundamentação teórica necessária, no Capítulo 3 é descrito em detalhes os experimentos aqui realizados, no Capítulo 4 é detalhado passo a passo o projeto, implementação, aplicação e coleta dos dados do aplicativo. Já no Capítulo 5 é avaliado os resultados e limitações deste trabalho.

2 Fundamentação

Este capítulo faz uma revisão dos conceitos de ES, arquitetura de software, narrativas interativas, desenvolvimento para dispositivos móveis e jogos sérios ou educativos e sua respectiva avaliação.

2.1 Engenharia de Software e seu Ensino

Para Pressman (2016) a Engenharia de Software (ES) consiste em: "Um arcabouço para as tarefas necessárias para construir softwares de alta qualidade." Com isso ela abrange um processo, um conjunto de métodos e práticas de ferramentas que possibilitam aos profissionais desenvolverem software de elevada qualidade. O autor também define a qualidade, como item preponderante para o sucesso de um software, e vários estudos determinam que ela esteja intrinsecamente ligada ao atendimento a requisitos. A camada de processo constitui o pilar para o controle e gerenciamento do trabalho de desenvolvimento. A camada de métodos oferece subsídios técnicos para as tarefas desenvolvidas durante o ciclo de desenvolvimento, e por último na camada mais alta temos as ferramentas fornecendo automação das atividades de forma organizada e repetível.

A engenharia de software ES fornece os subsídios necessários para a criação e manutenção de softwares. A área citada possui diversos campos para atuação tais como: Interação Humano-Computador, Processo de Software, Engenharia de Requisitos, Projeto de Software, Teste de Software, Gestão de projetos e Arquitetura de Software sendo pauta recorrente nos currículos de graduação em computação, apoiada nas Diretrizes Curriculares de Cursos da Área de Computação e Informática do Ministério da Educação (MEC). Há que se ressaltar sobre a assimilação dos conceitos e teorias relacionadas à engenharia de software, os quais recomendam que estes aprendizados sejam aplicados através da prática em laboratórios e estágios. Logo, existem pesquisas com bacharéis da área de Ciência da Computação (CC), que atuam no ramo de software, que confirmam a necessidade da junção entre a teoria e a prática com os tópicos de engenharia de software

depois da graduação, ou seja, as competências de engenharia de software não estão sendo adequadamente abordadas nos cursos de CC (SILVA; MÜLLER; BERNARDI, 2011).

A ES está relacionada à aplicação de teoria e prática para o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos dos usuários (ABRAN et al., 2004). De forma geral, o ensino dessa disciplina é disponibilizado como um elemento curricular obrigatório formal no nível de graduação, pós-graduação ou por meio de treinamentos profissionais de curta duração. Quanto ao modo de ensino, o currículo de ES Abran et al. (2004) destaca a possibilidade do desenvolvimento de técnicas de ensino que vão além de aulas expositivas e em laboratórios. Em substituição, poderiam acontecer discussões práticas de casos, dinâmicas em grupos e uso de jogos (PRIKLADNICKI et al., 2009).

A Sociedade Brasileira de Computação (SBC) ressalta dois currículos de referência, em que um relaciona-se aos cursos de Bacharelado em CC e a Engenharia de Computação (EC) e o outro para cursos de Bacharelado em Sistemas de Informação (SI) (JONATHAN, 2016). Na proposta da SBC para o curso de CC e EC as disciplinas organizam-se de duas formas: Fundamentos da Computação e Tecnologia da Informação. O processo de formação acadêmica engloba a disciplina ES, na qual destaca o teste de software em um aspecto, que é a verificação, validação e teste. No currículo do curso de SI, as disciplinas de computação dividem-se em três núcleos: Fundamentos da Computação e Sistemas de Informação e Formação Tecnológica.

2.2 Arquitetura de software

Arquitetura de Software consiste na especificação da estrutura fundamental de um sistema de software em um nível elevado de abstração (CLEMENTS, 2010). Os componentes dessa arquitetura, além de possuírem maior tamanho, são mais relevantes para os objetos a que se propõe (VALENTE, 2020). Dessa forma, Taylor e Dashofy (2009) define os padrões arquiteturais como em uma solução reutilizável para um problema recorrente em arquiteturas de software em um determinado contexto. Nesta seção, serão detalhados os seguintes padrões de arquiteturas: arquitetura em camadas; arquitetura MVC; arquitetura em microsserviços; e arquitetura orientada a mensagens.

2.2.1 Arquitetura em Camadas

Os primeiros modelos de sistemas de software de grande porte surgiram nas décadas de 60 e 70. O padrão seguido por esses sistemas seguem o padrão de camadas, dispostas de forma hierárquica. De acordo com suas utilidades, a arquitetura em camadas pode ser usada para protocolos de redes. Uma vantagem desse padrão arquitetural está no fato de decompor o sistema em componentes menores; em segundo lugar, ela pode disciplinar as dependências entre essas camadas (VALENTE, 2020).

Já a arquitetura em três camadas, por exemplo, é bastante utilizada no mercado corporativo, que executava sistema de mainframes até meados dos anos 1980. Com o avanço da tecnologia de redes e hardware, foi possível modificar para plataformas com três camadas, sendo elas: Interface do Usuário, Lógica de Negócio e Banco de Dados. A Figura 2.1 ilustra a interface que pode ser oferecida aos clientes.

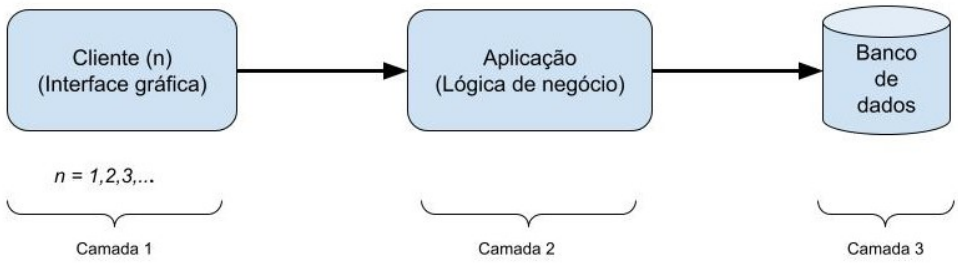


Figura 2.1: Diagrama de arquitetura em três camadas. Diagrama adaptado de Valente (2020).

2.2.2 Arquitetura em MVC

O padrão MVC surgiu nos anos de 1970 e foi um dos modelos pioneiros em linguagens orientadas a objetos, além de sugerir mudanças de interfaces gráficas como janelas, *scroll bars*, entre outros. Esse modelo foi escolhido como padrão para projetistas de *Smalltalk* na implementação de interfaces gráficas. O MVC define as classes de sistemas em três grupos, sendo eles: visão, controladoras e modelos (VALENTE, 2020).

Segundo Valente (2020) uma das vantagens da MVC pode ser descrita na contribuição para que classes de Modelo sejam utilizadas com diversas Visões e o favorecimento da testabilidade.

A cerne do padrão MVC está na separação entre código de interface com o

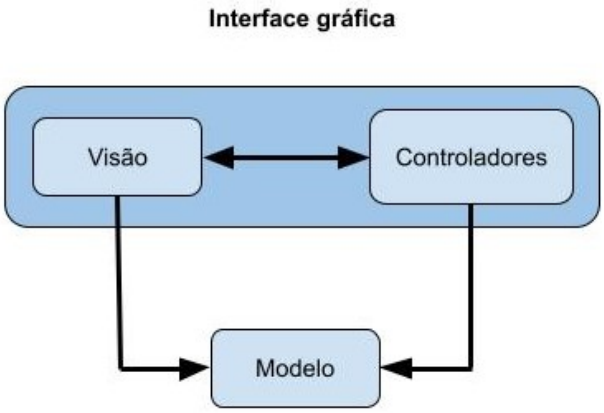


Figura 2.2: Classes de uma arquitetura MVC. Diagrama adaptado de Valente (2020).

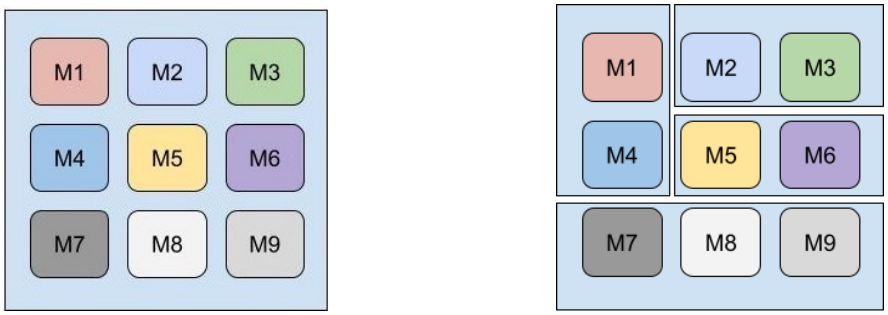
usuário (a Visão, também chamada de apresentação) e a lógica do domínio (o Modelo). As classes de Controle tratam e interpretam eventos gerados por dispositivos de entrada. Como resultado de tais eventos, Controladoras podem solicitar uma alteração no estado do Modelo ou da Visão. Por outro lado, objetos de domínio não incluem código visual, mas apenas lógica de negócios. Isso separa duas partes complexas de sistemas de software em partes que são mais fáceis de se modificar. Também permite várias apresentações da mesma lógica de negócio (FOWLER; BECK, 1999).

Com o surgimento de novas tecnologias, tais como a Internet e as aplicações web, novas arquiteturas como o MVC fizeram-se necessárias. Podendo fazer uso de *servlets* e JavaServer Pages (JSP) mesclados, temos sempre a figura de um *servlets* controlador que despacha as solicitações HyperText Transfer Protocol (HTTP) para as páginas de apresentação JSP. Além disso, devemos ter classes auxiliares para: realizar conexões com o banco de dados e atender solicitações de acesso ao mesmo; criar classes de domínio que refletem os principais objetos a serem tratados no domínio.

2.2.3 Arquitetura em Microsserviços

Sistemas que via de regra, seguem em tempo de execução de uma arquitetura monolítica, correm o risco de sobrecarga, ou seja, mesmo que o desenvolvimento tenha sido particionado em módulos M1, M2, M3, ..., Mn, em tempo de execução esses módulos são executados pelo sistema operacional como um processo único. Assim, todos os módulos compartilham o mesmo espaço de endereçamento (VALENTE, 2020). O tempo de execução

de um sistema pode ser comparado com um grande monólito, conforme ilustra a Figura 2.3a.



(a) Monólito com nove módulos. (b) Módulos isolados em microsserviços.

Figura 2.3: Aplicação monolítica em (a) convertida em microsserviços em (b). Diagrama adaptado de Valente (2020).

De forma que garanta aos seus clientes, a solução de *bugs* inesperados, as empresas que utilizam arquiteturas monolíticas têm um poderoso e rigoroso processo de lançamento de *releases*, que incluem testes manuais antes de concordarem com sua produção.

Para agilizar os serviços, diversas empresas adotaram a troca de monolíticos para uma arquitetura baseada em microsserviços, conforme destaca Valente (2020), como: certos grupos de módulos são executados em processos independentes (Figura 2.3b), sem compartilhamento de memória. Ou seja, o sistema é decomposto em módulos não apenas em tempo de desenvolvimento, mas também em tempo de execução. Com isso, as chances de que mudanças em um módulo causem problemas em outros módulos ficam bem menores. Quando os módulos são separados em processos distintos não há mais possibilidade de que um módulo acesse um recurso interno de outro módulo, como uma variável global, um atributo estático ou uma interface interna. Em vez disso, por construção, toda comunicação tem que ocorrer via interfaces públicas dos módulos. Assim, microsserviços são um instrumento para garantir que os times de desenvolvimento somente usem interfaces públicas de outros sistemas. A obediência à essa regra é garantida pelo sistema operacional.

2.2.4 Arquitetura Orientada a Mensagens

A comunicação entre os clientes e os servidores é mediada por um terceiro serviço que têm a única função de prover uma fila de mensagens, conforme a Figura 2.4

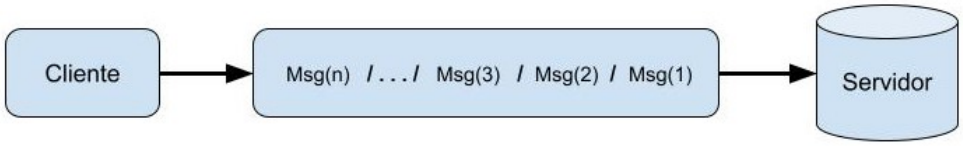


Figura 2.4: Arquitetura orientada a mensagens. Diagrama adaptado de Valente (2020).

Nesse ponto, os clientes é que atuam mais assertivamente como produtores de informação, quando inserem as mensagens nas filas. Os servidores retiram as mensagens e processam as informações que podem ser extraídas delas, consideradas registros com conjuntos de dados. A fila de mensagens estrutura-se através do *(First In, First Out)* (FIFO), cujo processo pode ser explicado como: a primeira mensagem que entra na fila é a primeira a ser consumida pelo servidor.

O uso de filas de mensagens faz com que a comunicação pelo lado do cliente seja assíncrona. Quando a informação é colocada na fila, o cliente pode continuar seu processamento. Assim, é importante que o serviço de mensagens seja instalado em equipamento eficiente e com poder de processamento altamente reconhecido. Também é importante que a fila de mensagens seja persistente. Se o servidor que regulariza a fila cair, os dados serão perdidos. Como as filas de mensagens são muito usadas na construção de sistemas distribuídos, existem soluções já prontas no mercado.

O padrão, além de permitir a comunicação assíncrona entre clientes e servidores, as filas de mensagens viabilizam duas formas de desacoplamento entre os componentes de uma aplicação distribuída, sendo elas: desacoplamento no espaço (clientes não precisam conhecer os servidores e vice-versa) e desacoplamento no tempo (clientes e servidores não precisam estar simultaneamente disponíveis para se comunicarem). Assim, as filas de mensagens permitem também escalar mais facilmente um sistema distribuído (VALENTE, 2020).

2.2.5 Outros Padrões Arquiteturais

O padrão *Pipes* e filtros é um tipo de arquitetura orientada a dados, na qual os programas, chamados de filtros, têm como função processar os dados recebidos na entrada e gerar uma nova saída. Os filtros são conectados por meio de *pipes*, que agem como *buffers*. Isto é, pipes são usados para armazenar a saída de um filtro, enquanto ela não é lida pelo próximo filtro da sequência. Com isso, os filtros não precisam conhecer seus antecessores e sucessores, o que torna esse tipo de arquitetura bastante flexível, permitindo as mais variadas combinações de programas. Além disso, por construção, filtros podem ser executados em paralelo (VALENTE, 2020).

A Cliente-Servidor é uma arquitetura muito usada na implementação de serviços básicos de rede. Clientes e servidores são os dois únicos módulos desse tipo de arquitetura e eles se comunicam por meio de uma rede. Os clientes solicitam serviços ao módulo servidor e aguardam o processamento. Já as arquiteturas *peer-to-peer* são aquelas cujos módulos da aplicação podem desempenhar tanto o papel de cliente, como o papel de servidor (módulos de pares) (VALENTE, 2020).

2.2.6 Anti-padrões Arquiteturais

O anti-padrão arquitetural é conhecido por ser um conceito não recomendado. O mais popular dentre os anti-padrões é chamado de *big ball of mud* (ou grande bola de lama, em uma tradução livre). Ele detalha sistemas nos quais qualquer módulo se relaciona com praticamente qualquer outro módulo (VALENTE, 2020).

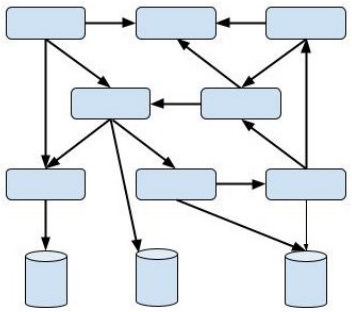


Figura 2.5: Anti-padrões Arquiteturais. Diagrama adaptado de Valente (2020).

2.3 Jogos, jogos sérios e gamificação

O senso popular afirma que o foco dos jogos é o entretenimento. Já os *serious games* são jogos cujo objetivo está relacionado à educação, ciência, cuidados de saúde, treinamentos de emergências, planejamento de cidades, engenharia e áreas militares.

O termo "jogos sérios" permeia seriedade e resolução de problemas com o experimental e a liberdade emocional do jogo. Os jogos sérios combinam a concentração analítica e questionadora do ponto de vista científico com a liberdade intuitiva e recompensa construtiva dos atos artísticos (ABT, 1970). Jogos sérios oferecem então um novo mecanismo para ensino e treinamento, combinando jogos digitais com educação. Logo, podem ir além das videoaulas e livros, de forma que garanta a demonstração e aplicação dos conhecimentos aprendidos. Porém, o uso de jogos sérios requer planejamento, pois não podem funcionar quando os jogadores não compreendem as regras, os objetivos no jogo, as consequências de suas ações e as razões para suas consequências (ABT, 1970).

Abt (1970) foi quem primeiro utilizou o termo *serious games*. Sua definição consiste em reduzir o jogo à sua essência formal: um jogo é uma atividade entre dois ou mais decisores independentes que procuram alcançar os seus objetivos em algum contexto limitante. Uma definição mais convencional diria que um jogo é um contexto com regras entre os adversários que tentam alcançar objetivos. Estamos preocupados com *serious games* no sentido de que estes jogos têm uma explícita e cuidadosa finalidade educativa pensada e não se destinam a serem jogados principalmente por diversão (ABT, 1970).

O ato de jogar relaciona-se com a relação do estudante com o sistema, que inclui as decisões e reações, comportamento e o *feedback* em relação ao ciclo, pois a interação do estudante com o jogo atinge objetivos educacionais. O governo brasileiro, em agosto de 2014, por meio do Ministério das Comunicações, investiu 4,5 milhões em *serious games* para estimular a indústria de games e tecnologia no Brasil como meio de estimular sua produção (PAULA, 2011).

Outra discussão para essa categoria de jogos seria: "O que separa *serious games* do restante dos jogos é o foco em um resultado de aprendizado específico e intencional para alcançar mudanças de desempenho e comportamento sérios, mensuráveis e continuados" (DERRYBERRY, 2007).

Em âmbito internacional, Paula (2011) afirma que os Estados Unidos são pioneiros no ramo: O jogo *America's Army*, desenvolvido para o exército americano e distribuído gratuitamente pela internet a partir de 2002, é considerado o primeiro *serious games* significativo de todo o mundo, com 17 milhões de downloads registrados somente em 2004 (PAULA, 2011).

2.4 Narrativas e Narrativas Interativas

Observando o crescimento dos jogos eletrônicos e do mercado internacional nessa área. É comum verificarmos que os jogos são classificados a partir do termo gênero. A classificação por gêneros, inicialmente tinha a intenção de assemelhar-se à classificação comercial como aquelas adotadas pelo mercado de livros e do cinema, uma vez observada a grande circulação. Dessa forma, o gênero literário pode se apresentar de duas maneiras: erudita ou comercial. Na classificação comercial, utilizada em livrarias, o gênero está associado à temática da literatura (SATO; CARDOSO, 2008).

Os alunos possuem grande facilidade para lidar com forma diferenciadas de ensino. Sob essa ótica, o *Northwestern University Knight Lab* criou ferramentas interativas gratuitas de narrativa de histórias, fazendo milhares de histórias *on-line* que designers e desenvolvedores de *eLearning* podem usar para atrair alunos enquanto apresentam informações em vários formatos e mídias (DESENHO. . . , 2017).

As ferramentas gratuitas mais relevantes na criação de narrativas interativas tais como: Inky[1], Twine [2], ScummVM [3] e Inform 7 [4]. As seguintes também gratuitas, foram desenvolvidas pelo *Northwestern University Knight Lab*, cada uma com sua particularidade: TimelineJS [5], StoryMapJS [6], StorylineJS [7] e Soundcite [8]. Destacam-se entre as que apresentam acesso a todas funcionalidades da aplicação somente mediante

[1]Editor para a linguagem da narrativas Ink disponível em ⟨https://www.inklestudios.com/ink⟩.
[2]⟨https://www.twinery.org⟩.
[3]⟨https://www.scummvm.org⟩.
[4]⟨http://inform7.com/⟩.
[5]Criação de narrativas através de uma *timeline*. ⟨https://timeline.knightlab.com/⟩.
[6]Focada na edição e anotações em um mapa ou imagem com informações históricas. ⟨https://storymap.knightlab.com/⟩.
[7]Permite ilustrar e explicar uma história de dados ao longo do tempo. ⟨https://storyline.knightlab.com/⟩.
[8]Adiciona sons, músicas ou palavras faladas às histórias. ⟨https://soundcite.knightlab.com/⟩.

pagamento: Klynt [9], Korsakow [10], RacontR [11], Dalet Pyramid [12], StoryKit [13], Genial [14], Thinglink [15], Pixton [16] e RichCast [17]

Ferramenta	Código Aberto	Gratuita	Gráfica	Editor	Dispositivos Móveis
Inky	✓	✓	-	✓	✓
Twine	✓	✓	✓	✓	-
ScummVM	✓	✓	✓	✓	✓
Inform 7	✓	✓	✓	✓	✓
TimelineJS	✓	✓	✓	✓	-
StoryMapJS	✓	✓	✓	✓	-
StorylineJS	✓	✓	✓	✓	-
Soundcite	✓	✓	✓	✓	-
Klynt	✓	-	✓	✓	✓
Korsakow	✓	-	✓	✓	-
RacontR	-	-	✓	✓	✓
Dalet Pyramid	-	-	✓	✓	-
StoryKit	-	-	✓	✓	✓
Genial	-	-	✓	✓	✓
Thinglink	-	-	✓	✓	✓
Pixton	-	-	✓	✓	-
RichCast	-	-	✓	✓	-

Tabela 2.1: Tabela comparativa entre ferramentas utilizadas na criação de narrativas interativas.

Dentre as ferramentas utilizadas para as narrativas interativas, buscamos destacar duas, em formato gratuito. São elas a *Twine* e *ScummVM*.

Segundo Rocha (2019), a ferramenta *Twine* surgiu em 2009 e obteve reconhecimento maior em 2012. Foi criado visando gerar visualizações de narrativas interativas com *hyperlinks* e texto em suas passagens, muito acessível e intuitiva para pessoas sem conhecimentos em programação. O *Twine* é baseado em *HyperText Markup Language* (HTML), com vários formatos de história, possibilitando o uso de *Cascading Style Sheets* (CSS) e *JavaScript* para personalizar e implementar as interações, layout e acesso a diferentes recursos multimídias. Os formatos de histórias foram incluídos no *Twine 2.0* e oferecem diferentes experiências para o desenvolvedor. Tais formatos interferem o layout de interação do usuário, e oferecem diferentes mecânicas pré-estabelecidas. Entre os formatos de histórias mais populares estão:

1. *Harlowe:* oferece uma aparência e interações formatadas para ser possível a criação de uma história sem conhecimentos em linguagens Web;

[9] ⟨https://www.klynt.net/⟩.
[10] ⟨http://www.korsakow.com/⟩.
[11] ⟨https://racontr.com/⟩.
[12] ⟨https://www.dalet.com/products/pyramid/⟩.
[13] ⟨https://storykit.io/⟩.
[14] ⟨https://genial.ly/⟩.
[15] ⟨https://www.thinglink.com/⟩.
[16] ⟨https://www.pixton.com/⟩.
[17] ⟨https://www.richcast.com/⟩.

2. *Snowman:* concede um layout mínimo, que exige sua customização em HTML e CSS;

3. *SugarCube:* apresenta recursos de layout, como menus para salvar e reiniciar o jogo, mas exige conhecimentos avançados de CSS e *JavaScript*.

ScummVM é uma aplicação que permite você executar os clássicos jogos de aventura em diversas plataformas, como *Windows*, *Linux*, *MacOS*, *iOS*, *PlayStation*, *Nintendo* e outros, além de comportar-se como emulador de jogos antigos no estilo "aponte e clique".

Dentre os *adventures* compatíveis com o *ScummVM* estão: *Simon the Sorcerer 1 e 2 da Adventure Soft; Beneath a Steel Sky, Broken Sword 1 e 2 da Revolution; Flight of the Amazon Queen; Inherit the Earth da Wyrmkeep; Gobliiins da Coktel Vision; The Legend of Kyrandia da Westwood Studios* e jogos baseados no tradicional sistema da *LucasArts*, o *Script Creation Utility for Maniac Mansion* (SCUMM) - que significa "Utilitário de Criação de Comandos para o Jogo *Maniac Mansion*", tais como *Indiana Jones, Monkey Island, Day of the Tentacle, Sam & Max* dentre outros.

2.5 Narrativas interativas para ensino de Engenharia de Software

Os jogos estão relacionados à realidade, buscando o ensinamento através do entretenimento. Incorporado nesse contexto, um jogo voltado para o ensino e aprendizagem precisa contemplar duas dimensões: ser um jogo, ou seja, haver uma competição com restrições para ao final se vencer; e ser educativo, que consiste em ser projetado para transferir determinado conhecimento (ABT, 1970).

Cursos de Mestrado em Administração de Negócios (MBA) já utilizam jogos do tipo *Business Game*, simulando negócios de forma dinâmica e divertida e com fins educacionais, em um ambiente livre de riscos reais (ANTUNES; PAIXãO; ALBUQUERQUE, 2019).

Apesar da popularidade dos jogos atravessar esse período de ampla aprovação

que engloba uma maior aceitação entre alunos e professores, existem alguns obstáculos que retardam sua aplicação e aproveitamento de suas potencialidades e que, segundo Wang (2005), foram os formadores dos três pilares fundamentais para o sucesso na utilização de jogos. São eles: educadores preparados, estrutura que permita a aplicação dos jogos e um planejamento adequado, outrossim, uma variedade e qualidade de jogos à disposição que se encaixam no planejamento (ANTUNES; PAIXãO; ALBUQUERQUE, 2019).

Será disponibilizado para os alunos da disciplina ES dois modelos de narrativas, um exemplo abordando os conceitos de Java Enterprise Edition (Java EE) e dois exemplos com o foco principal desse trabalho que é a Arquitetura de Software.

Vale salientar que durante a exploração dos estados das narrativas baseadas em arquitetura de software o jogador irá encontrar um estado com características que o difere dos outros, nele O jogador pode retornar repetidamente a um ponto (como um 'hub' em um jogo) a partir do qual pode fazer escolhas adicionais (NEIL, 2018). Serão apresentadas alternativas de rotas e, além disso, é necessário retornar repetidamente a esse ponto, definimos assim como *loops* ou *hubs*. Soma-se a isso a possibilidade de escolhas serem adicionadas ou eliminadas de acordo com o número de vezes que o jogador visita o *hub*. A imagem a seguir ilustra um exemplo de um *hub* genérico.

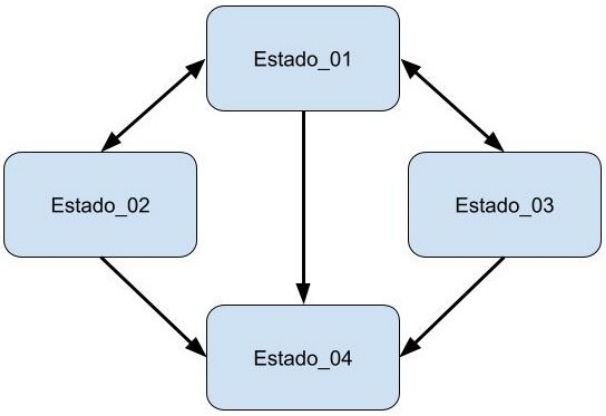


Figura 2.6: Exemplo de *hub* genérico. Diagrama adaptado de Neil (2018).

A narrativa interativa baseada nos padrões para camada de negócios em aplicações Java EE foi adaptada do trabalho "Aprendendo Padrões Java EE com uma História Interativa" publicado por Perillo (2009), apresenta uma situação onde um empregado em uma empresa precisa tomar decisões de design e longo da trajetória, conhecendo as consequências dessas ações.

2.6 Dispositivos móveis

Conforme o acesso a recursos de softwares aumentou, o desenvolvimento de aplicativos móveis na ES tornou-se uma ferramenta básica a ser criada pelos próprios estudantes.

No contexto apresentado que Lima K. M.; Violin (2019) desenvolveram um aplicativo para que estudantes de Engenharia Civil possam ampliar seus conhecimentos sobre as situações e conceitos que envolvem cada etapa de uma obra, além de incentivá-los a se tornarem alunos ativos no próprio processo de aprendizagem. Na mesma área, Dias (2018) desenvolveram um outro aplicativo para dispositivos Android responsável pela classificação de solos. Com parâmetros inicialmente obtidos de experimentos em campo, o usuário do aplicativo consegue obter o tipo de solo de acordo com os métodos *Transportation Research Board* (TRB) e *Unified Soil Classification System* (USC). Vaz (2019) apresenta os aplicativos de uma maneira mais lúdica ao desenvolver um aplicativo para o ensino de Engenharia de Produção por meio de um jogo. Enquanto Lamy (2018) produz em um aplicativo computacional para a função perda de qualidade, Silva (2019) trabalha no aperfeiçoamento de um aplicativo para dispositivo móvel orientado a gestão de projetos de recuperação de áreas degradadas. Por outro lado, Lima G. M.; Santos (2019) buscam utilizar um aplicativo móvel para celular no ensino do projeto de câmaras de separação gravitacional, onde como um dos resultados o usuário consegue um gráfico de eficiência de coleta conforme o diâmetro da amostra (DIAS et al., 2020).

É possível recorrer a ambiente fantasioso para tratar de um assunto sério, de forma que o jogador possa refletir sobre as possíveis situações reais. Também se nota que as simulações podem resgatar temas que não necessitam da iteração humana, como os simuladores de tráfego de automóveis, nos quais a simulação ocorre de forma independente e os participantes apenas observam (SILVA, 2016).

Este capítulo fez uma revisão dos conceitos fundamentais para o entendimento das contribuições deste trabalho. No próximo capítulo serão discutidas o método utilizado para a criação do protótipo de aplicativo que poderá facilitar o ensino de ES, o método da pesquisa será detalhado e um cronograma de atividades proposto.

3 Método

Este capítulo apresenta o método utilizado durante a realização deste trabalho.

Este trabalho de pesquisa exploratória, busca aplicar as ferramentas existentes para explorar o uso de narrativas interativas para ensino de arquitetura de software.

Como resultado final se espera a criação de um novo jogo sério capaz de simular as interações em uma empresa na qual o aluno tomará decisões relacionadas aos padrões arquiteturais, devendo identificar as características e uso de cada um.

As histórias serão compiladas em um aplicativo voltado para o ensino da ES. Através deste protótipo para Android, espera-se que gere engajamento com os múltiplos finais para depois ser assessoradas e acompanhadas pelo professor, que agirá como mediador dos estudantes com o conteúdo provido pela aplicação em paralelo com o conteúdo expositivo.

O método utilizado para a realização da criação das narrativas até o protótipo do aplicativo para *Android* foi subdividido nas seguintes etapas:

1. Definição de cenários para arquiteturas de software da literatura;
2. Investigação das principais e mais populares ferramentas para construção de histórias narrativas;
3. Adaptação de uma história interativa sobre Java EE em ambas nas ferramentas escolhidas;
4. Construção de duas novas histórias interativas para arquiteturas na ferramenta *Twine*;
5. Adaptação das narrativas para *Ink script*;
6. Identificação de como implementá-las no *Android*;
7. Implementação das histórias em um aplicativo para dispositivos móveis utilizando o *blade-ink*.

O processo de construção da ferramenta se deu a partir da revisão bibliográfica do conteúdo de Arquitetura de Software juntamente com a definição de um modelo com os principais aspectos e dinâmicas do assunto, produzindo assim duas narrativas interativas com este tema. Além dessas foi utilizado neste trabalho uma outra narrativa já existente baseada nos conceitos de Java EE. Posteriormente as três histórias interativas foram convertidas para as ferramentas *Twine* e em seguida para a *Inky*.

Com o aplicativo *Android* após ter passado por algumas etapas de refinamento e estando pronto para receber as narrativas foi então utilizada a biblioteca *blade-ink* que realizou a conversão, e as três histórias interativas se tornaram jogáveis no dispositivo móvel.

Este capítulo apresentou o método utilizado neste trabalho. O próximo capítulo detalha cada uma das etapas desenvolvidas e respectivos resultados.

4 Desenvolvimento

Neste capítulo é detalhado todo o desenvolvimento das narrativas interativas e suas regras, juntamente com a implementação do aplicativo no software *Android Studio* quanto das narrativas nas ferramentas *Twine* e mais especificamente na ferramenta *Inky* onde foi possível a integração com o *Android Studio* através da biblioteca *Blade Ink*.

4.1 Definição dos objetivos educacionais

O objetivo é fazer com que o aluno após passar pelas duas narrativas interativas, totalizando seis desafios-testes verifique como os seus conceitos teóricos sobre a disciplina de arquitetura de software estão de acordo com o que foi o seu aprendizado sobre cada modelo de projeto.

Os modelos de arquitetura apresentados em Valente (2020) foram todos abordados nas narrativas seja como o conceito principal de cada desafio, seja como alternativas que testarão os conceitos adquiridos pelo aluno, que são: Arquitetura MVC, Arquitetura em Camadas, Microsserviços, Arquiteturas Orientadas a Mensagens, *Pipes* e Filtros, Arquiteturas Publish/Subscribe, Cliente/Servidor e *Peer-to-peer*.

4.2 Construção das narrativas

Nas duas narrativas desenvolvidas para este trabalho envolvendo Arquitetura de software, sendo a primeira abordando como principal os conceitos de Arquitetura MVC, Arquitetura em Camadas e Microsserviços e a segunda narrativa com Arquitetura Orientada a Mensagens, *Publish/Subscribe* e *Peer-to-peer*; são apresentados exemplos de situações hipotéticas onde um candidato a uma vaga em uma determinada empresa precisa solucionar três situações problemas. Após escolher em qualquer ordem um dos desafios é apresentado uma situação em que o aluno deverá identificar qual o padrão de arquitetura (MVC, camadas, microsserviços. etc.) foi citado naquele desafio e assim escolher qual das alter-

nativas estão se referindo ao padrão correspondente. O jogador pode se deparar com duas ou três alternativas a cada estado intermediário que o levará até o final da história onde caso ele acerte todas as alternativas sobre o padrão de arquitetura daquele desafio, será assim recomendada a sua contratação pela empresa. Podem ainda haver um final não tão bom e um final ruim onde o jogador não conseguiu escolher nenhuma alternativa correta.

Para cada situação problema que o jogador poderá selecionar, foi escolhido um padrão de arquitetura principal que contém quatro estados abordando este conceito até chegar ao estado final correto. Já as outras alternativas são estados baseados em conceitos de padrões aleatórios que buscam induzir o aluno a desviar do caminho e completar o desafio de forma incorreta. Após finalizar um teste o mesmo não ficará mais disponível, posteriormente os três forem concluídos, ficará então disponível um estado para o jogador finalizar o jogo.

4.2.1 Implementação no Twine

Quando o jogador inicia a interação com a narrativa é apresentado a imagem com a topologia do grafo de forma ampla de uma das alternativas oferecidas sobre arquitetura de software com destaque para o *hub* que após o nó inicial é o estado no qual o jogador deverá escolher uma das três situações problema a serem solucionadas e após completar esse objetivo parcial o jogador faz o retorno para o mesmo. A captura de tela do *hub* da narrativa interativa em sua versão *web* apresenta ao jogador três possíveis caminhos sem ordem pré estabelecida até a solução final do problema pode ser vista na 4.1.

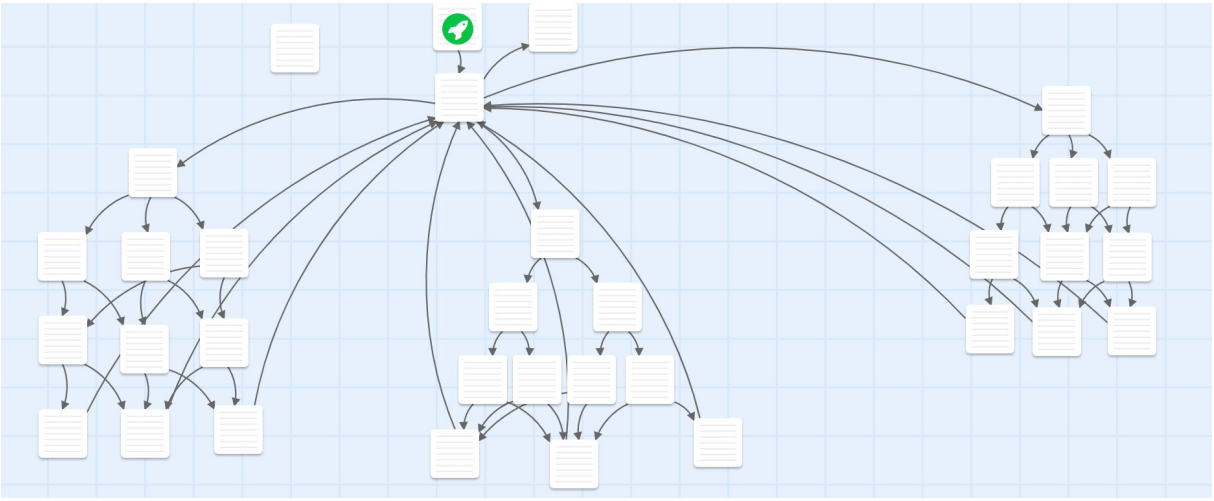


Figura 4.1: Topologia do grafo de uma narrativa produzida no *Twine*

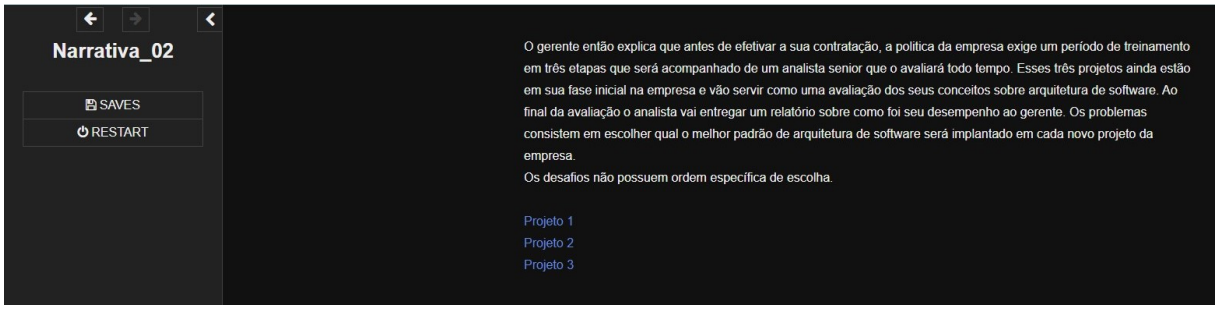


Figura 4.2: Narrativa interativa produzida no *Twine*

4.2.2 Implementação no Inky

Para ser feito a conversão das narrativas interativas em linguagem *Ink* foi utilizado o editor oficial chamado *Inky*. O detalhamento desse processo de conversão foi orientada seguindo os passos indicados pelo próprio manual oficial da linguagem que é o que os desenvolvedores aconselham.

Ao iniciar a ferramenta *Inky*, é apresentado uma janela com duas colunas. No lado esquerdo é onde se escreve o *script* da narrativa, e no lado direito é reproduzir uma prévia dele com as formatações próprias da linguagem. A partir daí ocorre o desenvolvimento da narrativa utilizando apenas texto escrito, com anotações especiais chamadas de "marcações" que tornam o texto interativo.

O cerne da linguagem é suficiente para a conversão tanto da narrativa baseada nos conceitos Java EE quanto nas narrativas envolvendo os conceitos de Arquitetura de Software, composto basicamente de estruturas definidas como: nós, desvios e escolhas.

Os nós são seções vinculadas, ou trechos da história que podemos visitar uma ou várias vezes. O início de um nó é indicado no *Inky* usando pelo menos dois sinais de igual o nome do nó do lado esquerdo e, opcionalmente, também do lado direito. O destaque em azul no *Inky* indica a marcação de tinta que foi reconhecida, confirmando para nós que foi escrita corretamente e fazendo com que ela se destaque como separada do nosso conteúdo.

Utilizamos desvios usando a seta própria para essa função que possui uma sintaxe própria no *Ink* (entrada como "menos" e depois "colchete"), contamos a história para ir para um nó diferente. Quando a história for jogada, a junção será automática e

invisível para o jogador. Para tratar as escolhas que o jogador toma, a linguagem *Ink* oferece várias formas de faze-la. Nesse trabalho foi utilizado uma sintaxe, após o sinal de "+" o texto se torna clicável para o jogador e consequentemente coloca-se a seta do desvio indicando para qual nó aquela decisão levará o jogador.

Para a criação de um estado específico, definido como "Opcao_01", com a finalidade de se comportar como um *hub* foi utilizado a marcação "*" ao invés de "+" no início das alternativas a serem escolhidas para que as mesmas não estejam mais disponíveis na próxima vez que o jogador visitar o *hub* após finalizar cada um dos problemas propostos.

A Listagem 4.1 apresenta como são testados se o jogador realmente chegou até o final de cada desafio, ou seja, se ele visitou qualquer um dos nós finais do primeiro desafio e qualquer um dos nós finais do segundo desafio e qualquer um dos nós finais do terceiro desafio, para somente assim estar disponibilizado o estado "Fim" que encerra o jogo e como pode ser visto na imagem abaixo.

Listagem 4.1: Verificação de visita dos estados finais

```
''{(Opcao_02_07 or Opcao_02_08 or Opcao_02_09) and
(Opcao_03_07 or Opcao_03_08 or Opcao_03_09) and
(Opcao_04_07 or Opcao_04_08 or Opcao_04_09)}
Finalizar -> Fim''
```

```
=== Opcao_01 ===
O gerente então explica que antes de efetivar a sua contratação, a politica da empresa exige um período de
    treinamento em três etapas que será acompanhado de um analista senior que o avaliará todo tempo. Esses
    três projetos ainda estão em sua fase inicial na empresa e vão servir como uma avaliação dos seus
    conceitos sobre arquitetura de software. Ao final da avaliação o analista vai entregar um relatório sobre
    como foi seu desempenho ao gerente. Os problemas consistem em escolher qual o melhor padrão de
    arquitetura de software será implantado em cada novo projeto da empresa.
Os desafios não possuem ordem específica de escolha.
*Projeto 1 ->Opcao_02
*Projeto 2 ->Opcao_03
*Projeto 3 ->Opcao_04
*{(Opcao_02_07 or Opcao_02_08 or Opcao_02_09) and (Opcao_03_07 or Opcao_03_08 or Opcao_03_09) and
    (Opcao_04_07 or Opcao_04_08 or Opcao_04_09)} Finalizar ->Fim
```

Figura 4.3: Código do *hub* na narrativa produzida no editor *Inky*

Todavia, para a implementação do mesmo *hub*, na sintaxe que o *Twine* adota foram observadas algumas particularidades que serão ilustradas na captura de tela a seguir. Cada um dos projetos está vinculado a uma variável do tipo *boolean*, previamente inicializada com o valor "false". Ao término do desafio a variável troca o seu conteúdo para "true" consequentemente executando o comando *else* e tornando indisponível o problema

que já fora escolhido anteriormente. Para a liberação do estado “Fim” um teste é aplicado fazendo-se necessário que as três variáveis contenham valores “true”.

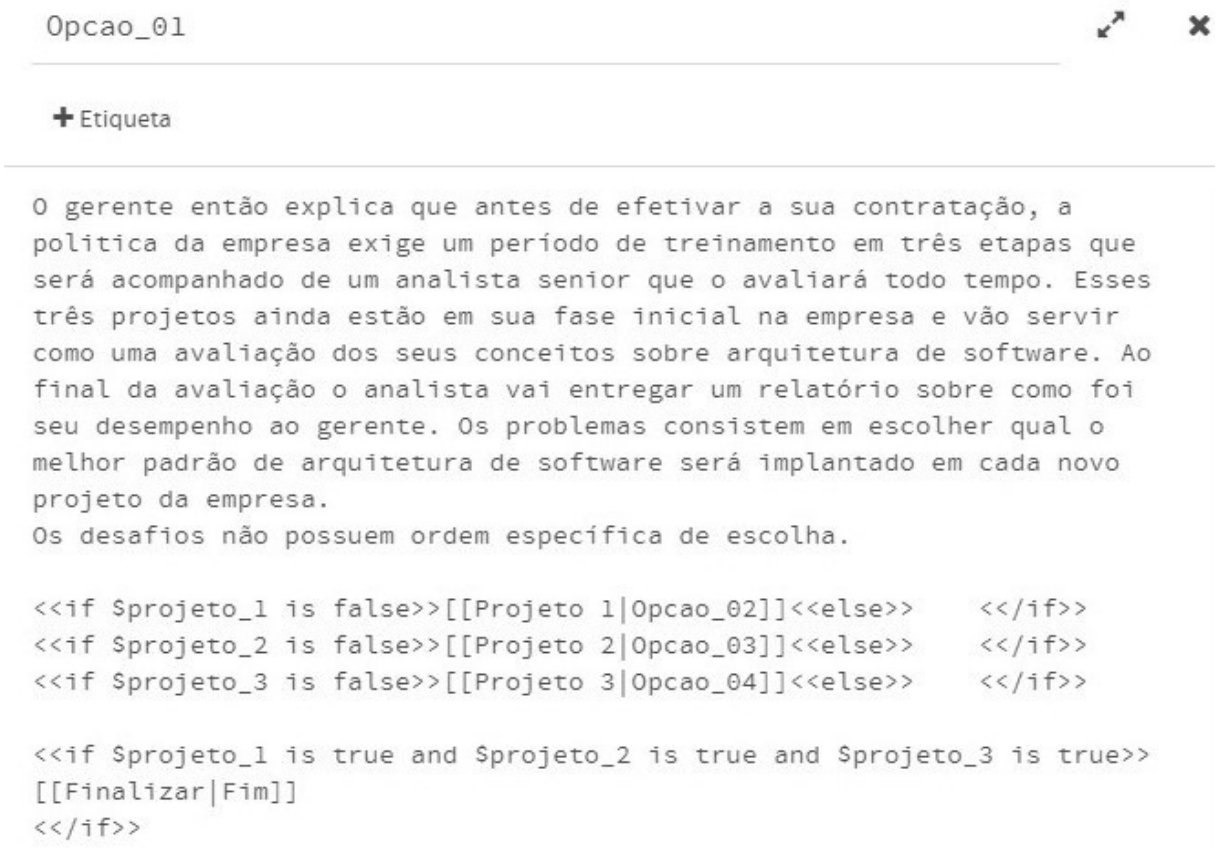


Figura 4.4: Código do *hub* na narrativa produzida no editor *Twine*

Neste trabalho foram utilizadas as ferramentas de criação de narrativas interativas: o *Twine* e o *Inky*, cada uma com suas particularidades. No que se refere ao *Twine*, seu diferencial está na possibilidade de visualização no formato de grafo de estados da narrativa em si, facilitando muito o trabalho do desenvolvedor, pois ele pode ter a visão rápida e completa de toda estrutura da narrativa com seus nós e desvios.

Contudo no *Inky* essa possibilidade de visualização gráfica esquematizada não é possível, porém sua virtude está no fato do que tornou esse trabalho possível. Com esse editor pode-se gerar arquivos do tipo “.json” daí torna-se possível a integração com sistema *Android* através da biblioteca *Blade Ink*.

4.3 Desenvolvimento do Aplicativo em Android

Inicialmente na versão *Arctic Fox* e após uma atualização a versão *Bumblebee* do ambiente *Android Studio*, o primeiro passo foi o desenvolvimento de um protótipo apenas com uma tela de seleção com as opções: "Jogar" onde o jogador poderá escolher entre os três exemplos de narrativas, "Instruções" com um breve tutorial do jogo, "Créditos" com o reconhecimento feito pelo aluno e seu orientador e "Sair" para encerrar o aplicativo.

Após a finalização das narrativas interativas, foi iniciada a etapa de migração para o aplicativo com o auxilio da biblioteca *Blade Ink* que reconhece a narrativa criada no software *Ink* através de uma exportação do arquivo no formato.json. Foi adicionado um *TextView* para o texto principal da história e mais três objetos com texto com cores diferentes para as alternativas que são clicáveis. Decorrida a fase dedicada à correção de erros, a ferramenta reconheceu a narrativa interativa e foi possível a exibição do texto principal quanto das alternativas.

Foi então inserida uma tela de "boas vindas" com animação, uma funcionalidade de enviar *e-mail* para os desenvolvedores deste aplicativo através do um click na tela, foi inserido mensagens de confirmação ao sair do aplicativo e também houve correções de outros erros.

Iniciando o aplicativo "ES Interativa" o aluno tem a opção de jogar e testar os seus conhecimentos, com isso ele será direcionado à tela de seleção de narrativas na qual ele poderá escolher entre três opções, sendo as duas primeiras opções de narrativas baseadas nos conceitos de Arquitetura de software e a terceira, um exemplo demonstrativo dos conceitos de Java EE, além de um botão "Voltar" para reler o menu anterior conforme apresentado nas figuras 4.5 e 4.6.

Conforme a evolução no jogo, o aluno irá se deparar com situações onde ele deverá ler o texto proposto para aquele estado, arrastar a tela do celular até o final quando necessário e escolher com um toque o melhor caminho para o próximo estado como pode-se observar nas figuras 4.7 e 4.8. Concluído os três desafios, é apresentado ao jogador uma mensagem de encerramento do jogo com os dizeres "Obrigado por ter jogado".

Pode se ser traçado um paralelo da versão *Android* com as versões *web* do *Twine* e *Inky*. Sem sombras de dúvidas, o principal diferencial está na mobilidade que a

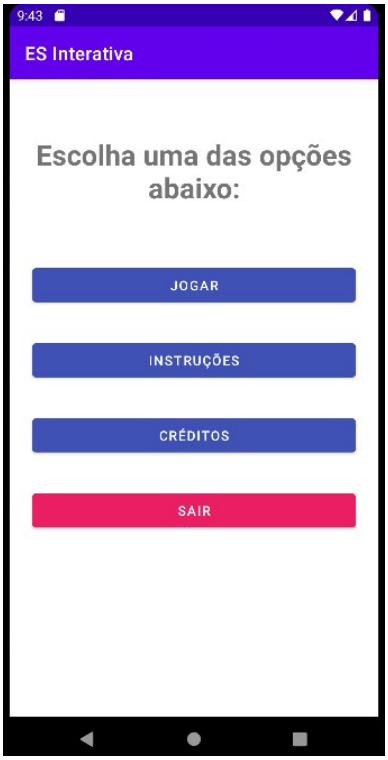


Figura 4.5: Tela de seleção de opções

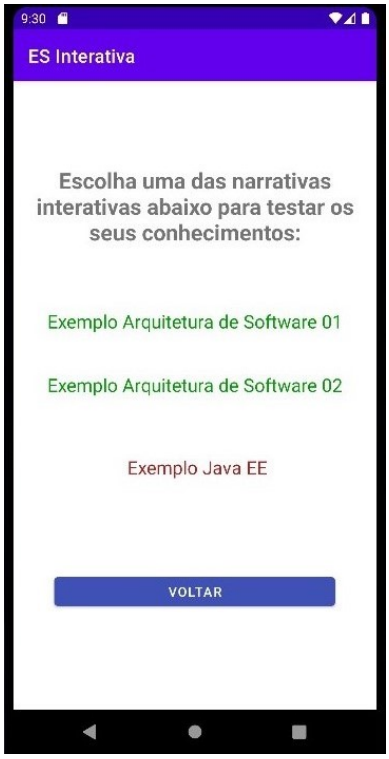


Figura 4.6: Tela de seleção de histórias

versão *Android* apresenta tornando assim possível sua utilização tanto em salas de aula, auditórios ou até mesmo em campo aberto durante alguma atividade que o professor propor e onde não é possível a utilização de computadores ou até mesmo localidades remotas que não possuem uma rede elétrica acessível.

Este capítulo apresentou como se deu as etapas do desenvolvimento utilizado neste trabalho. O próximo capítulo detalha as considerações finais, limitações e os trabalhos futuros sugeridos.

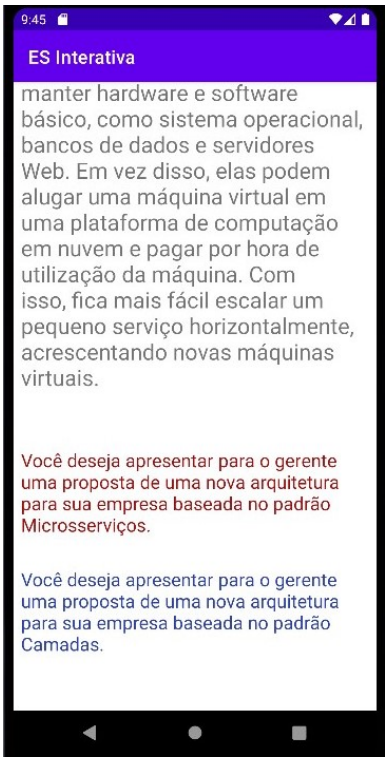


Figura 4.7: Primeira tela de transição

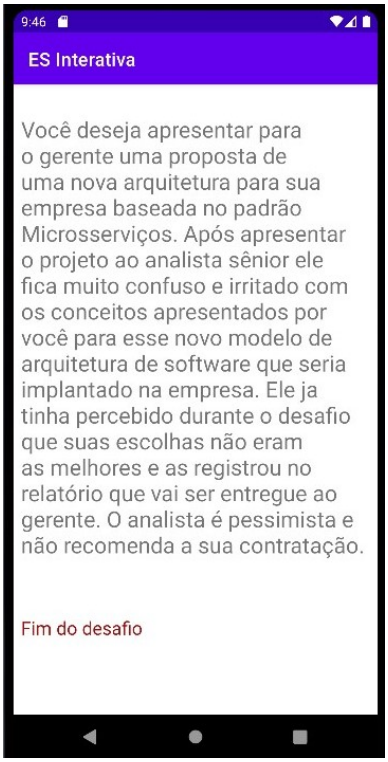


Figura 4.8: Segunda tela de transição

5 Considerações Finais

O problema enfrentado corresponde à dificuldade de trazer formas alternativas ao ensino acadêmico, mais especificamente ao aprendizado da Arquitetura de Software e, por consequência, um maior engajamento dos alunos.

Faz-se ressaltar que esse engajamento dos alunos de engenharia de software no curso é importante e necessário, dada a ausência de métodos alternativos e interativos para tal ensino. Assim, as narrativas interativas são uma excelente alternativa, segundo estudos pesquisados.

Um aspecto que deve ser destacado é a criação de um aplicativo educacional, voltado para as aulas, pode contribuir para diminuir o distanciamento entre os docentes e seus alunos no que diz respeito ao acesso de técnicas de aprendizado através de jogos digitais, fazendo com que os discentes desfrutem mais intensamente do potencial dos jogos na educação.

Foi realizada uma pesquisa sobre o uso de softwares no ensino da ES e como ocorrem as construções das narrativas interativas em um contexto educacional depois foram convertidas no formato dos softwares *Twine* e *Inky*.

O problema foi atacado a partir da criação de duas narrativas interativas no software *Twine*, então foram convertidas à linguagem *Ink* para poderem ser integradas através da biblioteca *Blade Ink* a um aplicativo desenvolvido no ambiente *Android Studio*, tornando mais simples a sua utilização por alunos através de um aparelho celular.

O objetivo geral foi utilizar, através da criação de um protótipo digital, histórias interativas como um jogo sério para os conteúdos sobre arquitetura de software. Para atingi-lo, os seguintes objetivos específicos foram: fazer uma revisão da bibliografia de arquitetura de software, definir um modelo com os principais aspectos e dinâmicas do assunto; montar um conjunto de situações problema que reproduzem situações e efeitos de ações de projeto. Esses objetivos foram atingidos com sucesso.

Já o objetivo: avaliar a aplicação das narrativas interativas para o ensino de arquitetura de software não foi alcançado, pois houve um tempo excessivo utilizado

no desenvolvimento e correção de erros do aplicativo, com isso foi ultrapassado o prazo previsto inicialmente de envio para o mesmo fosse colocado em teste por alunos para pudessem dar o seu feedback.

Dessa forma, esse trabalho é uma contribuição para o inicio de uma ferramenta que atuará no auxilio do ensino do curso de engenharia de software podendo ser melhor aproveitado através de experiências que aproximem o ensino da realidade do aluno. Embora seja trabalhosa a montagem de um jogo de narrativas interativas, cabe destacar a facilidade da utilização de jogos no aplicativo *Android*, no software *Twine* e convertidas à linguagem *Ink*, visto a facilidade de acesso por parte dos alunos em seus smartphones.

A próxima etapa é concentrar mais no desenvolvimento e avaliação do aplicativo. Em um trabalho futuro poderá ser feita uma avaliação mais completa do trabalho, se possível com turmas de alunos de diferentes instituições pois assim com esse *feedback* sendo captado de situações totalmente heterogêneas o aplicativo tornará mais robusto e com novas funcionalidades que os avaliadores irão sugerir.

Podem ser citadas algumas simplificações que foram adotadas para a realização deste trabalho, tais como: desenvolvimento do aplicativo em uma plataforma *IOS*, fazendo assim sua utilização muito mais ampla entre os alunos e professores, verificando então como as histórias interativas podem contribuir no ensino de arquitetura de software; utilização de um *layout* mais amigável, com mais cores, mais efeitos de animação, sons e imagens exercendo um maior interesse nas narrativas, logo um maior engajamento dos alunos nas disciplinas de engenharia de software de uma maneira geral; um maior número de narrativas interativas poderiam ser produzidas, se possível por diferentes autores e fossem reunidas no mesmo aplicativo, assim os alunos teriam contato com diferentes formas de abordagem de um mesmo tópico de uma determinada disciplina.

Uma possível forma alternativa de abordar o problema de forma promissora, seria a utilização das narrativas interativas como jogos sérios no formato *web* mais tradicional, seria perdido o fator principal, a mobilidade, que o *smartphone* oferece já citado anteriormente, porém com a utilização de monitores maiores, acarreta uma melhora na leitura dos textos e na visualização de detalhes das narrativas interativas.

Bibliografia

ABRAN, A. et al. Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*, 2004.

ABT, C. C. *Serious Games*. [S.l.]: New York: Viking, 1970.

AMéLIO, C. de O. A indÚstria e o mercado de jogos digitais no brasil - evolução, características e desafios. Universidade Federal de Minas Gerais, 2018.

ANTUNES, K. D. d. N.; PAIXãO, G. S.; ALBUQUERQUE, M. C. N. Mapeamento sistemático de jogos digitais para o ensino-aprendizagem. 2019.

CARVALHO, V. Baixo investimento em ciência e tecnologia eleva a desigualdade social. 2020. Acessado em in 18/02/2022. Disponível em: ⟨https://www.ufg.br/n/129177-baixo-investimento-em-ciencia-e-tecnologia-eleva-a-desigualdade-social⟩.

CHEN, W.-F. et al. Work in progress - a game-based learning system for software engineering education. *2008 38th Annual Frontiers in Education Conference*, p. T2A–12–T2A–13, 2008.

CLEMENTS, P. F. B. L. B. D. G. J. I. R. L. P. M. R. N. J. S. *Documenting Software Architectures: Views and Beyond, Second Edition*. [S.l.]: Addison-Wesley Professional, 2010.

DERRYBERRY, A. "serious games: online games for learning". i'm serious.net. 2007.

DESENHO Instrucional. 2017. ⟨https://www.desenhoinstrucional.com/post/ferramentas-interativas-gratuitas-para-storytelling⟩. Acessado em 29/07/2021.

DEUS, L. Jogos podem auxiliar no aprendizado. 2016. Acessado em in 18/02/2022. Disponível em: ⟨https://revistaeducacao.com.br/2016/05/10/aprendizado-em-jogo/⟩.

DIAS, C. et al. O aprendizado de algoritmos como ferramenta do engenheiro do amanhã: Desenvolvimento de aplicativos móveis para instigar o uso da computação. In: . [S.l.: s.n.], 2020.

DIAS, C. A. R. e. a. Promovendo o ensino em engenharia por meio da interdisciplinaridade: desenvolvimento de um software para o ensino de mecânica dos solos. in: Xlvi congresso brasileiro de educação em engenharia e 1º simpósio internacional de educação em engenharia, 2018, salvador (ba). anais. 2018.

ENRIQUEZ, S. M. O. D. Previsões da idc brasil para 2021 apontam que mercado de tic crescerá 7%. 2021. Acessado em in 18/02/2022. Disponível em: ⟨https://www.idc.com/getdoc.jsp?containerId=prLA47452221⟩.

FOWLER, M.; BECK, K. *Refactoring: improving the design of existing code*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc, 1999.

JONATHAN, M. *Currículos de Computação: porque permanecem assim? - XXXVI Congresso da Sociedade Brasileira de Computação*. 2016. ⟨www.https://www.sbc.org.br/⟩. Acessado em 19/02/2022.

LAMY, L. Aplicativo computacional para a função perda de qualidade, razão sinal-ruído e análise experimental de taguchi. 2018. 50 f. dissertação (mestrado) - universidade federal do paraná, curitiba. 2018.

LIMA G. M.; SANTOS, K. D. O uso de um aplicativo móvel para celular no ensino do projeto de câmaras de separação gravitacional. in: Xxxix congresso brasileiro de sistemas particulados, 2019, belém-pa. anais. 2019.

LIMA K. M.; VIOLIN, R. Y. T. Gamificação aplicada ao processo de aprendizagem no curso de engenharia civil. in: Xi encontro internacional de produção científica, 2019, maringá-pr. anais. 2019.

NEIL, K. Authoring interactive narrative in Twine 2 vs Ink vs Yarn. 2018. Cited in 12/02/2022. Disponível em: ⟨https://medium.com/@haikus_by_KN/e695eb4dfc3e⟩.

PAULA, G. N. D. A prática de jogar videogame como um novo letramento. dissertação (mestrado em linguística aplicada). instituto de estudos da linguagem. universidade estadual de campinas. 2011.

PERILLO, E. G. R. Aprendendo padrões java ee com uma história interativa. *MundoJ*, 2009.

PRESSMAN, R. S. Engenharia de software, uma abordagem profissional–8ª ed–amgh editora ltda. *Porto Alegre–RS-2016*, 2016.

PRIKLADNICKI, R. et al. Ensino de engenharia de software: Desafios, estratégias de ensino e lições aprendidas. in: Fórum de educação em engenharia de software. 2009.

ROCHA, W. C. e Carla Rodriguez e Denise Goya e Mirtha Venero e R. Software livre twine: ensino de programação web por meio da criação de jogos educacionais. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, v. 8, n. 1, 2019.

SANTOS, R. dos et al. Utilizando experimentação para apoiar a pesquisa em educação em engenharia de software no brasil. In: . [S.l.: s.n.], 2008.

SATO, A. K. O.; CARDOSO, M. V. Além do gênero: uma possibilidade para a classificação de jogos. 2008.

SILVA, J. G. d. Plataforma para criação de jogos educativos para usuários não-experientes. dissertação (mestrado) – universidade federal de pernambuco. cin, ciência da computação, recife. 2016.

SILVA, R. D. C. Aperfeiçoamento de aplicativo para dispositivo móvel orientado a gestão de projetos de recuperação de áreas degradadas. 2019. tese (graduação) - universidade federal de mato grosso, mato grosso. 2019.

SILVA, T. T. D.; MÜLLER, F. M.; BERNARDI, G. Panorama do ensino de engenharia de software em cursos de graduação focado em teste de software: Uma proposta de aprendizagem baseada em jogos. In: . [S.l.: s.n.], 2011.

TAYLOR, N. M. R. N.; DASHOFY, E. M. *Software Architecture: Foundations, Theory, and Practice*. [S.l.]: Wiley, 2009.

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade*. [S.l.]: Moderna, 2020.

VAZ, A. C. d. S. B. Desenvolvimento de um jogo digital para ensino de engenharia de produção. 2018. 50 f. tese (graduação) - universidade federal de campina grande, paraíba. 2019.

WANG, W. O aprendizado através de jogos para computador: por uma escola mais divertida e mais eficiente. são paulo, 2005. 2005.