Universidade Federal do Piauí
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação

Uma biblioteca para manipulação de dados contextuais em jogos pervasivos móveis

Vitor Augusto Correa Cortez Almeida

Número de Ordem PPGCC: M055

Teresina–PI, Abril de 2018

Vitor Augusto Correa Cortez Almeida

Uma biblioteca para manipulação de dados contextuais em jogos pervasivos móveis

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Universidade Federal do Piauí – UFPI

Centro de Ciências da Natureza

Programa de Pós-Graduação em Ciência da Computação

Orientador: Ricardo de Andrade Lira Rabêlo

Teresina–PI

Abril de 2018

Vitor Augusto Correa Cortez Almeida
Uma biblioteca para manipulação de dados contextuais em jogos pervasivos móveis/ Vitor Augusto Correa Cortez Almeida. – Teresina–PI, Abril de 2018-
81 p. : il. (algumas color.) ; 30 cm.

Orientador: Ricardo de Andrade Lira Rabêlo

Dissertação (Mestrado) – Universidade Federal do Piauí – UFPI
Centro de Ciências da Natureza
Programa de Pós-Graduação em Ciência da Computação, Abril de 2018.

1. Biblioteca. 2. Jogos pervasivos. I. Ricardo de Andrade Lira Rabêlo. II. Universidade Federal do Piauí – UFPI. IV. Uma biblioteca para manipulação de dados contextuais em jogos pervasivos móveis

CDU 02:141:005.7

Vitor Augusto Correa Cortez Almeida

Uma biblioteca para manipulação de dados contextuais em jogos pervasivos móveis

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da UFPI (área de concentração: Sistemas de Computação), como parte dos requisitos necessários para a obtenção do Título de Mestre em Ciência da Computação.

Trabalho _____.

Ricardo de Andrade Lira Rabêlo
Presidente da Banca Examinadora

José Valdemir dos Reis Junior
Examinador Interno

Windson Viana de Carvalho
Examinador Externo

Fabbio Anderson Silva Borges
Examinador Externo

Teresina–PI
Abril de 2018

Dedicada aos meus pais, Solano Cortez e Ivana Correa,
pois sem o seu suporte e sacrifícios, nada disso seria possível.

Agradecimentos

Primeiramente, muito obrigado à minha família. Em especial a minha mãe Ivana Correa, o meu pai Solano Cortez, e os meus irmãos, Joana Darc e Francisco Solano. É um privilégio vos ter ao meu lado, pois jamais seria possível concluir esse desafio, ou nem mesmo iniciá-lo, sem o vosso apoio incondicional, altruísmo e cuidado.

Muito obrigado ao meu orientador, o professor Ricardo Lira, por me direcionar na construção deste trabalho. A sua confiança foi o que me permitiu tomar minhas próprias decisões sobre o rumo da pesquisa e, embora eu não tenha sempre feito a melhor decisão, acredito que aprendi muito sobre como ser um bom pesquisador. Obrigado por todos os ensinamentos, por estar sempre presente, pela paciência, por não ter desacreditado de mim e pelo bom gosto em pizzas. Espero podermos colaborar em futuras pesquisas.

Obrigado aos meus professores da graduação que me incentivaram a iniciar o mestrado. Especialmente, os professores Raimundo Neto, Jeovane Reges e meu amigo Dann Luciano. Sempre tive em mente o quanto vocês acreditaram na minha capacidade, mesmo antes do início dessa jornada, e isso me serviu como grande motivação.

Agradeço a todos os alunos e professores do Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) da Universidade Federal do Ceará (UFC), que compartilharam comigo seus conhecimentos e me ajudaram a definir minha pesquisa. Obrigado em particular aos professores Fernando Trinta, Windson Viana, Luís Fernando e José Ricardo, e aos colegas Italo Linhares e Thalisson Oliveira.

Agradeço também a todos os colegas de mestrado e aos amigos que conheci no Laboratório de Otimização, Soluções Autônomas e Sistemas Inteligentes (OASIS) da Universidade Federal do Piauí (UFPI). Obrigado a Enza Rafaela, Douglas Lopes, Sidiney Araujo, Sávio Mota, ao professor *Marvin* Lemos, e a todos que contribuíram para a concretização desta pesquisa de maneira direta ou indireta.

Por fim, obrigado à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro concedido para a realização desta pesquisa.

"Remember that not getting what you want
is sometimes a wonderful stroke of luck".
(Harriett Jackson Brown Jr.)

Resumo

Um jogo pervasivo móvel pode ser definido como uma aplicação sensível ao contexto direcionada para plataformas móveis como, por exemplo, *smartphones*. Consequentemente, o desenvolvimento de jogos desse gênero apresenta desafios técnicos relacionados aos domínios das aplicações móveis e da computação sensível ao contexto. A possível existência de erros, imprecisões e incertezas nos dados contextuais usados pelas aplicações sensíveis ao contexto é uma limitação inerente dos sensores empregados para realizar medições. Por esse motivo, o desenvolvedor de aplicações sensíveis ao contexto precisa lidar com a incerteza apresentada pelos valores sensoreados para assegurar a estabilidade do comportamento da aplicação. Este trabalho apresenta como proposta uma biblioteca voltada para a manipulação dos dados contextuais no desenvolvimento de jogos pervasivos móveis. A biblioteca é baseada em um modelo conceitual orientado a objetos que define um conjunto de entidades para representar as informações contextuais ao longo dos processos de aquisição e interpretação do contexto determinados pelo modelo. Especificamente, essa biblioteca deve ser capaz de encapsular e abstrair detalhes do processamento de dados contextuais, dividir em módulos as tarefas de aquisição, tratamento e utilização do contexto, e oportunizar a reutilização de código referente à manipulação e a representação dos dados contextuais. As contribuições desta pesquisa são: a formulação de um modelo conceitual para representar dados contextuais; a especificação e implementação de uma biblioteca baseada no modelo proposto para auxiliar o desenvolvimento de jogos pervasivos móveis, capaz de aplicar diferentes técnicas de preprocessamento como sistemas de inferência *fuzzy* para lidar com incertezas; e a execução de um estudo experimental, de acordo com recomendações da Engenharia de *Software* Experimental, para avaliar como a biblioteca proposta pode beneficiar desenvolvedores de jogos pervasivos móveis. Os resultados do experimento indicam que soluções desenvolvidas com o apoio da biblioteca são menos extensas e mais simples, porém, não foram encontradas diferenças significantes no tempo de desenvolvimento e na percepção de qualidade. A biblioteca proposta por este trabalho foi intitulada Gamepad.

Palavras-chaves: aplicação sensível ao contexto, biblioteca de desenvolvimento, dados contextuais, jogos pervasivos móveis, tratamento de incertezas.

Abstract

A pervasive mobile game can be defined as a context-aware application directed towards mobile platform devices, such as smartphones. Consequently, the development process of these games presents technical challenges related to the domains of mobile applications and context-aware computing applications. The possibility of errors, imprecisions and general uncertainty among the contextual data employed by context-aware applications is an inherent limitation of the sensor devices used for measuring. Therefore, the developer of a context-aware application needs to handle the uncertainty presented by sensor data in order to ensure a stable behaviour for the application. This work proposes a software library to handle contextual data in the development of pervasive mobile games. The library is based on an object oriented conceptual model composed by a set of entities, which represent the contextual information across the model defined processes of acquiring and interpreting context. Specifically, this library must be capable of encapsulate and abstract contextual data processing details, separate the concerns of acquisition, handling and utilization of context, and facilitate code reuse. The contributions of this research are summarized as: the formulation of a conceptual model to represent contextual data; the specification and implementation of a library based on the proposed model to support the development of pervasive mobile games, able to apply distinct data preprocessing techniques such as fuzzy inference systems to handle uncertainty; and the execution of an experimental study, complying with recommendations of Software Engineering Experimentation, to evaluate how the proposed library can benefit pervasive mobile game developers. The results of the experiment suggest that solutions developed with the library are less verbose and more simple, however, no significant differences were found in relation to the development time and quality perception. The library proposed in this research was titled Gamepad.

Keywords: context-aware application, software library, contextual data, mobile pervasive games, uncertainty handling.

Lista de ilustrações

Lista de tabelas

Lista de listagens

Lista de abreviaturas e siglas

API	*Application Programming Interface*
CAC	*Context-Aware Computing*
CAS	*Context-Aware System*
FIS	*Fuzzy Inference System*
GPS	*Global Positioning System*
HAL	*Hardware Abstraction Layer*
JPM	Jogos Pervasivos Móveis
SDK	*Software Development Kit*
UbiComp	*Ubiquitous Computing*

Sumário

Introdução

A noção de computação ubíqua (UbiComp, *Ubiquitous Computing*), popularizada por Weiser (1991), descreve um modelo de interação humano-computador no qual as tecnologias da informação encontram-se integradas com o ambiente natural do ser humano. As tecnologias ubíquas desaparecem em plano de fundo, isto é, elas inserem-se no ambiente físico de forma transparente, tornando-se invisíveis para os usuários (WEISER, 1991). Essa descrição inspirou uma classe de aplicações que são cientes do contexto no qual estão a ser executadas e são capazes de responder a mudanças nos seus contextos para o benefício do usuário. Esta forma de computação, formada por dispositivos heterogêneos que buscam prover acesso ubíquo a esse tipo de aplicação, é denominada Computação Sensível ao Contexto (CAC, *Context-Aware Computing*) (SCHILIT; ADAMS; WANT, 1994).

Para Dey (2001), o "contexto" pode ser definido como toda informação a respeito de entidades que são consideradas úteis para caracterizar a interação entre aplicação e usuários. Uma vez que o próprio usuário é uma entidade relevante nesta interação, exemplos de informações contextuais seriam a sua posição geográfica, temperatura ambiente, a atividade física realizada, os recursos computacionais na proximidade, os interesses do usuário e até mesmo os seus sentimentos. Parte dessa informação é obtida por meio de sensores, enquanto a outra é inferida por meio de interpretações a partir de dados mais simples (ALEGRE; AUGUSTO; CLARK, 2016). Esta propriedade de usar o contexto como entrada primária é a principal característica dos Sistemas Cientes do Contexto (CAS, *Context-Aware Systems*).

Entre as diversas áreas que aplicam os conhecimentos promovidos pela computação ubíqua, encontra-se o desenvolvimento de jogos digitas. Desta união, dão-se o nome de jogos ubíquos ou pervasivos àquelas aplicações que, simplificadamente, fazem uso do contexto a fim de incrementar a interatividade com o jogador e agregar valor ao entretenimento (BECAM; NENONEN, 2008). O termo "pervasivo" no contexto dos jogos apresenta diversas interpretações na literatura (NIEUWDORP, 2007), porém, neste trabalho o termo é usado como sinônimo de *jogos ubíquos* (BJÖRK et al., 2002). Para Montola (2005), este gênero de jogo pode ser visto como uma versão estendida dos jogos convencionais, com um ou mais elementos que extrapolam limites temporais, espaciais e sociais.

Historicamente, a evolução dos jogos pervasivos têm acompanhado os avanços das tecnologias de comunicação, que tornam-se cada vez mais pervasivas. Na medida que surgem novas formas de interação entre ser humano e computador, novos tipos de jogos pervasivos podem ser criados — por exemplo, a atual geração de jogos pervasivos teve origem graças aos avanços das tecnologia móvel, que eliminou a necessidade dos jogadores de carregarem sensores externos (KASAPAKIS; GAVALAS, 2015).

Atualmente, os dispositivos móveis são grandes motivadores para a popularização e adoção dos jogos pervasivos. O sucesso de títulos como *Pokémon GO* mostram que os jogos pervasivos são atraentes para a população e apresentam-se como uma nova oportunidade comercial (PAAVILAINEN et al., 2017). Entre os benefícios das plataformas móveis, encontra-se a presença de sensores embutidos que permitem a captura direta de dados contextuais, como a movimentação do usuário ou a sua localização por meio do GPS (*Global Positioning System*). Essa característica possibilita a construção de JPMs (Jogos Pervasivos Móveis) sem a necessidade de investir em uma infraestrutura para suportar o acesso e a utilização dos dados contextuais e ainda conseguir proporcionar uma experiência de jogo satisfatória, interessante e divertida para o jogador.

Definição do Problema

Do ponto de vista técnico, parte do processo de desenvolvimento dos jogos pervasivos móveis pode ser comparada ao de uma aplicação sensível ao contexto direcionada para plataformas móveis como, por exemplo, *smartphones*. Consequentemente, o desenvolvimento desses jogos apresenta desafios técnicos relacionados aos domínios das aplicações móveis e sistemas sensíveis ao contexto. As tecnologias de comunicação, os tipos de contexto, a forma de representação visual dos dados contextuais, entre outros, são tópicos do design de jogos pervasivos frequentemente abordados na literatura (KASAPAKIS; GAVALAS, 2015). Além disso, a capacidade de lidar com dados contextuais diante de imprecisões e incertezas é um aspecto importante para os sistemas cientes do contexto, porque a existência de erros nessas informações pode prejudicar a aplicação.

A possibilidade da existência de erros e imprecisões em dados contextuais é uma limitação inerente dos sensores utilizados para realizar medições. De modo geral, o desenvolvedor de uma aplicação sensível ao contexto precisa estar ciente dos riscos envolvidos em utilizar dados obtidos por sensores, e ser capaz de preparar a aplicação para resistir a falhas presentes nesses dados (BETTINI et al., 2010). A maioria dos jogos pervasivos encontrados na literatura aborda esse problema por meio de soluções criativas aplicadas ao *design* do jogo, ou seja, modificando as regras do jogo para evitar ou amenizar efeitos das incertezas (VALENTE; FEIJÓ; LEITE, 2013). Por exemplo, em Robert-Bouchard, Dupire e Cubaud (2015 apud CHALMERS et al., 2004, p. 2) os jogadores são informados sobre a força do sinal de GPS e são incentivados a esconderem-se em localidades com sinal fraco para surpreender outros jogadores. No entanto, esse tipo de solução também pode ser visto como uma limitação, por que requer a adição de uma mecânica de jogo, o que pode não ser preferível na visão dos *designer*s do jogo.

Diversas abordagens para o tratamento de incertezas são encontradas na área de sensibilidade ao contexto. Geralmente, as soluções apresentadas visam oferecer suporte

a aplicações distribuídas e descentralizadas, por meio de infraestruturas de comunicação robustas (BALDAUF; DUSTDAR; ROSENBERG, 2007). Por outro lado, aplicações sensíveis ao contexto móveis frequentemente apresentam arquiteturas mais centralizadas, nas quais o dispositivo móvel é a principal fonte de dados contextuais (YÜRÜR et al., 2016). Por esse motivo, a utilização de soluções voltadas para aplicações sensíveis ao contexto em jogos pervasivos móveis JPMs necessita de uma etapa de adaptação para adequar a abordagem ao dispositivo móvel.

Dessa forma, percebe-se que as pesquisas em sensibilidade ao contexto visam aplicações voltadas para ambientes ubíquos e, por isso, é evidente que essas abordagens para a utilização do contexto não são direcionadas às particularidades dos JPMs. Assim, esse trabalho traz a seguinte pergunta de pesquisa: como melhorar o processo de manipulação dos dados contextuais no desenvolvimento de jogos pervasivos móveis? Como hipótese de pesquisa, acredita-se que abstrair a manipulação do contexto por meio de metáforas específicas do domínio de jogos permite ao desenvolvedor focar-se na lógica do jogo.

Visão Geral da Proposta

A proposta deste trabalho consiste numa biblioteca para a manipulação de dados contextuais em jogos pervasivos móveis que atua como uma mediadora entre a plataforma móvel e a lógica do jogo. A biblioteca é baseada num modelo conceitual orientado a objetos, cuja abstração é apresentada pela Figura 1. Tal modelo define quatro entidades — fontes, botões, interpretadores e controle, que simbolizam elementos dos controles de *video-game* tradicionais — para representar e manipular o contexto antes que este possa ser utilizado como entrada pela lógica do jogo. Tal processo de manipulação permite a utilização de diferentes técnicas de preprocessamento de entradas na forma de interpretadores, inclusive de sistemas inteligentes, como os sistemas de inferência *fuzzy*.

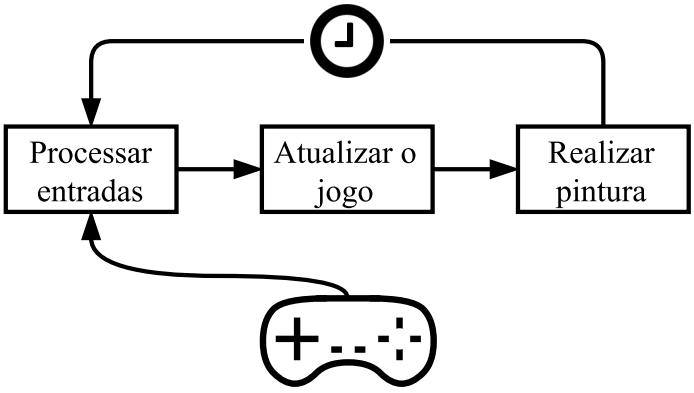


Figura 1 – Abstração empregada pelo modelo conceitual proposto.

Para ilustrar o tipo de jogo objetivado pela biblioteca, a subseção seguinte descreve um cenário no qual um jogo do gênero caça-ao-tesouro utiliza o contexto como entrada.

Cenário motivador

Juliana possui no seu *smartphone* um jogo de caça-ao-tesouro. Recentemente, ela soube que o jogo recebeu uma nova versão pervasiva e decide testá-la. De início, a interface do jogo não parece diferente da versão original, porém, Juliana percebe que a sua personagem não responde aos comandos de toque que funcionavam no jogo original.

Agora, o jogo pede para Juliana caminhar e ela percebe que quando caminha, a personagem move-se junto. Após caminhar por alguns segundos, Juliana avista um tesouro no mapa e corre em sua direção. Para a surpresa de Juliana, o tesouro também muda de posição, distanciando-se dela na mesma velocidade que sua personagem se movimenta.

Cansada de perseguir o objeto, Juliana desacelera e começa a caminhar devagar. Nesse momento, ela percebe que o tesouro para de se movimentar, como se não percebesse sua presença. Juliana entende a situação e aproveita a oportunidade, movendo-se vagarosamente até o tesouro. Eventualmente, sua personagem consegue capturá-lo, e logo em seguida mais dois objetos aparecem no mapa. Agora, um dos objetos aparenta se distanciar sempre que Juliana move o seu *smartphone*. Mais um enigma a ser resolvido.

Objetivos

Com o propósito de validar a hipótese de pesquisa apresentada, o objetivo geral desse trabalho é propor uma biblioteca de desenvolvimento de jogos pervasivos móveis para melhorar a manipulação de dados contextuais com base em uma abstração direcionada para esse domínio. Especificamente, essa biblioteca deve ser capaz de encapsular e abstrair detalhes do processamento de dados contextuais, separar a lógica das etapas de aquisição, tratamento e utilização do contexto e, de maneira geral, permitir o reúso de código relacionado à representação e manipulação desses dados.

Para atingir esse objetivo, são apontados os seguintes objetivos específicos:

- Identificar os requisitos das aplicações sensíveis ao contexto, convencionais e voltadas para dispositivos móveis, na literatura;
- Definir um modelo conceitual de referência, com base nos requisitos levantados, para abstrair o processamento de dados contextuais em JPMs;
- Implementar uma versão do modelo conceitual na forma de uma biblioteca de desenvolvimento de *software* para uma plataforma móvel específica;
- Avaliar e verificar, por meio de provas de conceito e estudos de caso, a usabilidade e eficácia da biblioteca implementada e a aplicabilidade do modelo de referência.

Justificativa

A abordagem proposta consiste em utilizar uma biblioteca, baseada em um modelo conceitual, para abstrair o tratamento de dados contextuais no desenvolvimento de jogos pervasivos móveis. O tratamento de dados contextuais em sistemas sensíveis ao contexto é um processo importante — conferindo tolerância a falhas e garantindo a integridade da aplicação — para o qual diferentes abordagens são exploradas atualmente (ALEGRE; AUGUSTO; CLARK, 2016). Como o tratamento do contexto é apenas uma das várias tarefas envolvidas no desenvolvimento de jogos pervasivos, decidiu-se propor uma biblioteca, cuja baixa complexidade permite uma maior flexibilidade em comparação com, por exemplo, um *framework*, que tem em foco apenas facilitar a interação com o contexto.

De modo a proporcionar uma solução genérica, que não esteja atrelada a uma única plataforma ou linguagem de programação, decidiu-se utilizar um modelo conceitual baseado em uma abstração para o domínio desses jogos. Abstrair um domínio complexo por meio de estruturas ou metáforas relacionadas ao domínio é uma forma de reduzir a complexidade e facilitar a compreensão dos desenvolvedores sobre as estruturas empregadas e sobre quais elementos são realmente relevantes (GARLAN; SHAW, 1994). Além disso, uma arquitetura quando voltada especificamente para uma estrutura de dados conhecida, nesse caso os dados contextuais, é possível tirar proveito desse conhecimento, levando em consideração a forma como esses dados são recebidos e a quantidade de fontes de dados, para oferecer abstrações mais adequadas que omitam funcionalidades desnecessários, ou seja, trocando flexibilidade por especificidade (DEY et al., 1999).

Contribuições

A seguir, são enumeradas as principais contribuições atribuídas a esta pesquisa:

1. Uma abstração e modelo conceitual para representar o contexto em JPMs, com a finalidade de aproximar o processo de manipulação dos dados contextuais do processamento de entradas utilizado em jogos digitais convencionais;

2. A especificação e implementação de uma biblioteca, com base no modelo proposto, que auxilia o desenvolvimento de JPMs com formalização dos processos de manipulação dos dados sensoreados e do tratamento de incertezas com o uso de técnicas genéricas, como os sistemas de inferência *fuzzy*;

3. Uma avaliação experimental empírica da biblioteca apresentada, a fim de verificar aspectos como sua usabilidade, flexibilidade em implementações, facilidade de entendimento e o cumprimento de seus requisitos funcionais.

Produção Científica

A seguir estão os trabalhos publicados que constituem esta pesquisa:

- ALMEIDA, Vitor A. C. C.; VIANA, José R. M.; RABÊLO, Ricardo A. L. "Designing pervasive game elements to handle uncertainty using fuzzy logic systems". In: *Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web*. Teresina, PI, Brasil: SBC, 2016. (WebMedia '16). v. 2, p. 137–140;

  i. Este trabalho apresentou uma abordagem inicial para o tratamento de incertezas utilizando motores de inferência *fuzzy* no domínio dos jogos pervasivos;

  ii. A abordagem proposta motivou a investigação de um modelo arquitetural para jogos pervasivos com suporte ao tratamento de incertezas.

- ALMEIDA, Vitor A. C. C.; RABÊLO, Ricardo A. L.; VIANA, José R. M.; MAIA, Luís F. "A model based on fuzzy control systems to support the development of pervasive mobile games". In: *International Conference on Systems, Man, and Cybernetics*. Banff, AB, Canada: IEEE, 2017. (SMC '17).

  i. Nesta pesquisa, o trabalho anterior foi amadurecido com a concepção de um modelo arquitetural para jogos pervasivos móveis baseados em regras *fuzzy*;

  ii. Com base em críticas e sugestões, o modelo arquitetural proposto foi reformulado, dando origem ao modelo conceitual apresentado pelo presente trabalho.

Estrutura do Trabalho

O Capítulo 1 apresenta o embasamento teórico deste trabalho, contextualizando as aplicações sensíveis ao contexto dentro da computação ubíqua e discutindo conceitos relacionados aos jogos pervasivos no seu processo de *design* e desenvolvimento. No final deste capítulo, são apresentados trabalhos no estado-da-arte sobre processamento do contexto, ressaltando os desafios da manipulação desse tipo de dado nas pesquisas relacionadas.

No Capítulo 2 são apresentados os requisitos para o tratamento de dados contextuais e jogos pervasivos móveis, e são detalhados os componentes do modelo utilizado pela biblioteca proposta. Em seguida, o Capítulo 3 descreve o processo de implementação de um jogo pervasivo móvel como prova de conceito para avaliar a biblioteca proposta.

Por fim, no Capítulo 4 o planejamento do estudo experimental conduzido para avaliar a biblioteca e seus resultados são discutidos, e o Capítulo 5 apresenta as conclusões deste estudo e as possibilidades de trabalhos futuros. Adicionalmente, os materiais utilizados na condução do experimento e análise dos resultados encontram-se nos Apêndices A e B respectivamente, seguidos de um plano experimental revisado no Apêndice C.

1 Fundamentação Teórica

Este capítulo apresenta os temas e conteúdos relacionados à abordagem proposta. Primeiramente, a computação sensível ao contexto é contextualizada e as principais características das aplicações sensíveis ao contexto são detalhadas. Em seguida, a área de pesquisa dos jogos pervasivos é detalhada e os desafios do desenvolvimento das bibliotecas de *software* e os sistemas de inferência *fuzzy* são apresentados. Por fim, um conjunto de trabalhos relacionados a esta pesquisa são discutidos.

1.1 Computação Sensível ao Contexto

A computação sensível ao contexto (do inglês, *context-aware computing*) é um conceito relacionado com a visão do pesquisador Mark Weiser sobre o futuro da computação, após observar a influência dos dispositivos computacionais e das tecnologias de comunicação no cotidiano das pessoas. Para Weiser (1991), o próximo passo evolutivo da computação estaria além dos computadores de mesa, por meio de uma grande diversidade de dispositivos computacionais que passariam a permear os ambientes físicos, e disponibilizar novas formas de interação, recursos e serviços de maneira transparente. Essa visão recebe o nome de "computação ubíqua" (WEISER, 1993).

O desafio fundamental da computação sensível ao contexto é conceder aos computadores a capacidade de reconhecer em qual situação, ou *contexto*, encontram-se seus usuários e utilizar esse conhecimento em aplicações e serviços a favor desses usuários (SCHILIT; ADAMS; WANT, 1994). Nas conversações entre seres humanos, o contexto é uma informação compreendida de forma implícita. No entanto, esse tipo de informação não é transmitido na interação convencional entre humanos e computadores, na qual são utilizados dispositivos de entrada tradicionais, como teclado e *mouse*. Por isso, para que a interação entre seres humanos e computadores seja enriquecida, é importante proporcionar a esses dispositivos os meios de acesso a informações contextuais (DEY et al., 1999).

Por meio de uma definição clara e concisa do quê constitui o contexto, desenvolvedores de aplicações podem decidir como representar essas informações situacionais computacionalmente. Dessa forma, é importante entender o que é contexto antes de poder utilizá-lo. Na literatura, a definição do contexto mais utilizada foi dada por Dey:

> Contexto são quaisquer informações que podem ser utilizadas para caracterizar a situação de uma entidade. Entidade é uma pessoa, lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo si mesmos (DEY, 2001).

Com base nessa definição, as informações que fazem parte do contexto de uma aplicação

ficam sob decisão dos desenvolvedores, de modo que o contexto para uma aplicação não é necessariamente igual ao de outra. Por exemplo, a temperatura ambiente de uma sala pode fazer parte do contexto de uma aplicação para o monitoramento do uso de ar-condicionados, mas a temperatura fora da sala não é do contexto.

Logo, as aplicações que pertencem ao cenário da computação sensível ao contexto precisam ser *sensíveis ao contexto* (do inglês, *context-aware application*). Isso implica que as aplicações devem ser capazes de utilizar as informações dos seus respectivos contextos em benefício dos seus usuários. Segundo Dey (2001), essas aplicações podem suportar três tipos de funcionalidades: (*i*) apresentação de informações e serviços; (*ii*) execução automática de serviços; e (*iii*) rotulação de informações contextuais para uso posterior. Sobretudo, a dinamicidade do contexto exige desse tipo de aplicação a capacidade de adaptar-se em resposta às novas informações contextuais. Aqui, ressalta-se uma importante característica das aplicações sensíveis ao contexto, isto é, a necessidade de lidar com dados de natureza contextual (ALEGRE; AUGUSTO; CLARK, 2016).

Dados contextuais não possuem um formato homogêneo, ou seja, diferentes estruturas de dados produzidas por fontes heterogêneas podem compor o contexto de uma aplicação ciente do contexto. Considerando a definição do contexto, uma observação válida sobre as informações contextuais é que grande parte delas são formadas por dados coletados a partir de dispositivos sensores (SANCHEZ et al., 2006; ZIEGLER et al., 2009). Assim como quaisquer formas de medição física estão sujeitas a imprecisões, esses dispositivos são susceptíveis a interferências que podem prejudicar a acurácia de suas medidas. Logo, antes que os valores sensoreados possam ser utilizados por uma aplicação ciente do contexto, é necessário que eles sejam analisados e interpretados, levando em consideração suas imperfeições e a possibilidade de incertezas (PERERA et al., 2014).

1.2 Jogos Pervasivos Móveis

Não existe um consenso na literatura sobre como definir os jogos pervasivos. Devido à natureza interdisciplinar do desenvolvimento de jogos e a ambiguidade do termo *pervasivo* em diferentes áreas, a definição do que é um "jogo pervasivo" passa a ser respondida sob diferentes perspectivas (NIEUWDORP, 2007). No estudo de Valente, Feijó e Leite (2017), pesquisas sobre jogos pervasivos foram classificadas em dois grupos: culturais e tecnológicas. No primeiro grupo, os jogos pervasivos são abordados com relação a questões de *design*, do *gameplay* e estudos sociais; já no segundo grupo, são enfatizadas as tecnologias que suportam a construção desses jogos. Por meio dessa divisão, é possível entender melhor o significado dos jogos pervasivos em cada perspectiva (VALENTE; FEIJÓ; LEITE, 2017).

Na perspectiva cultural, os jogos pervasivos são vistos como uma versão expandida das atividades lúdicas tradicionais. O termo "pervasivo" é utilizado para indicar que esses

jogos, em alguns aspectos, vão além das definições tradicionais dos jogos (VALENTE; FEIJÓ; LEITE, 2013). Um conceito bastante difundido é a ideia do "círculo mágico", que descreve o contexto ou um lugar especial no qual um jogo acontece, onde as regras do mundo real são esquecidas e substituídas pelas regras do jogo (SALEN; ZIMMERMAN, 2003). Segundo Montola (2005), os jogos pervasivos são aqueles que expandem o círculo mágico com base em três aspectos, ou dimensões: espacial, temporal e social. A dimensão espacial indica que o jogo pervasivo não possui um local físico fixo, na dimensão temporal o jogo pervasivo não tem uma hora para ser jogado ou hora para acabar, e a dimensão social indica que os participantes do jogo podem não saber que estão jogando (MONTOLA; STENROS; WAERN, 2009).

Na perspectiva tecnológica, os jogos pervasivos são vistos como aplicações da computação sensível ao contexto, e também são chamados de "jogos ubíquos" (BJÖRK et al., 2002). Os estudos classificados nessa categoria são divididos em duas abordagens (VALENTE; FEIJÓ; LEITE, 2013): (*i*) concepção de jogos "aumentados" por computador, ou seja, jogos do mundo real que são "melhorados" com o uso da tecnologia; e (*ii*) aplicações sensíveis ao contexto, com foco em questões sobre computação pervasiva/ubíqua e sensores. Para Linner et al. (2005), os jogos pervasivos são uma nova forma de aplicação sensível ao contexto caraterizados pelo foco no entretenimento e provisionamento multimídia. Logo, quando a plataforma alvo desses jogos é restringida para dispositivos móveis, como *smartphones*, eles são chamados de jogos pervasivos móveis.

Assim como outras aplicações sensíveis ao contexto, os jogos pervasivos móveis precisam lidar com incertezas. Na literatura, são identificadas cinco estratégias para lidar com incerteza (VALENTE; FEIJÓ; LEITE, 2017): remover, esconder, manipular, revelar e explorar. Enquanto a primeira estratégia consiste em evitar incertezas inteiramente na etapa de *design* do jogo, as duas estratégias seguintes reconhecem a possibilidade de incertezas e lidam com problemas em tempo de execução. Já as duas últimas estratégias consistem em aceitar a presença de incertezas ao ponto de incorporá-las no *gameplay*.

1.3 Interfaces de Programação da Aplicação

No desenvolvimento de *software*, incrementar o nível de abstração é uma atividade essencial para esconder a complexidade da pilha de tecnologias utilizadas por sistemas e permitir a construção de *software* cada vez mais complexos e robustos (GARLAN; SHAW, 1994). Uma forma comum de abstração são as bibliotecas de *software*, que representam um conjunto reutilizável de soluções (sub-rotinas) para um problema ou uma classe de problemas relacionados.

Um elemento importante de toda biblioteca é a sua API (*Application Programming Interface*), isto é, o conjunto de símbolos que representam as funcionalidades da biblioteca

e são disponibilizados para os seus usuários (BLANCHETTE, 2008). É por meio de APIs que as funcionalidades da biblioteca serão apresentadas aos desenvolvedores na construção de novas aplicações. Logo, da mesma forma que uma boa API facilita o trabalho do desenvolvedor, uma API ruim dificulta o desenvolvimento e tem o potencial de prejudicar todos os sistemas construídos nos níveis mais altos de abstração (HENNING, 2009).

Diversos trabalhos apontam aspectos que dizem respeito à usabilidade de uma API, como a facilidade de aprendizagem, reusabilidade e compreensão das abstrações empregadas. No trabalho de Robillard (2009), são apontados diversos problemas que dificultam a adoção de APIs, como a falta de recursos e documentação para a aprendizagem da API ou a utilização de *design*s complexos. Em Piccioni, Furia e Meyer (2013), os autores apresentam um estudo experimental para avaliar uma API, voltada para o acesso a banco de dados, que foi conduzido com foco na usabilidade. O estudo utilizou práticas de implementação que requeriam do participante descrever em voz alta sua linha de pensamento, e um conjunto de perguntas que foram aplicadas na forma de entrevista. Diversos problemas de *design* foram apontados com a execução do estudo, mostrando-se como um método eficaz e rigoroso para a avaliação da usabilidade de APIs. Outros trabalhos apontam linhas gerais ou qualidades de uma boa API. Para Myers e Stylos (2016), uma avaliação de usabilidade do *design* de uma API pode ser realizado com base em diferentes aspectos, como a nomenclatura das classes e métodos e se estes correspondem a objetos do mundo real, a consistência desses elementos e outros como assinaturas de métodos, *design* minimalístico e flexível, e a existência de documentações bem detalhadas.

1.4 Sistemas de Inferência *Fuzzy*

A teoria de conjuntos *fuzzy* é uma generalização da teoria de conjuntos clássica, na qual a pertinência dos elementos em relação a um dado conjunto (denominado conjunto *fuzzy*) pode assumir um valor parcial entre a pertinência total (1) e a não-pertinência (0) ao conjunto *fuzzy* (ZADEH, 1965). A lógica *fuzzy* é construída com base na teoria de conjuntos *fuzzy* e, de forma análoga, generaliza a lógica proposicional clássica para contemplar proposições com valores-verdade parciais entre a verdade total (1, verdadeiro) e a não-verdade (0, falso) (ZADEH, 1975).

O ser humano é capaz de resolver problemas complexos utilizando informações imprecisas e vagas, representadas por termos linguísticos (e.g. *baixa* temperatura, *alta* velocidade). A lógica *fuzzy* aliada à teoria de conjuntos *fuzzy* oferece um ferramental matemático para expressar e manipular informações imprecisas e nebulosas (DUBOIS; PRADE, 1980). Desde que seja possível representar a solução de um problema por meio de um conjunto de regras lógicas (*se...então*), é possível utilizar conjuntos *fuzzy* e a lógica *fuzzy* para expressar e manipular informações incertas com o uso dos sistemas de inferência

fuzzy, cujos componentes são ilustrados na Figura 2 (PEDRYCZ; GOMIDE, 2007).

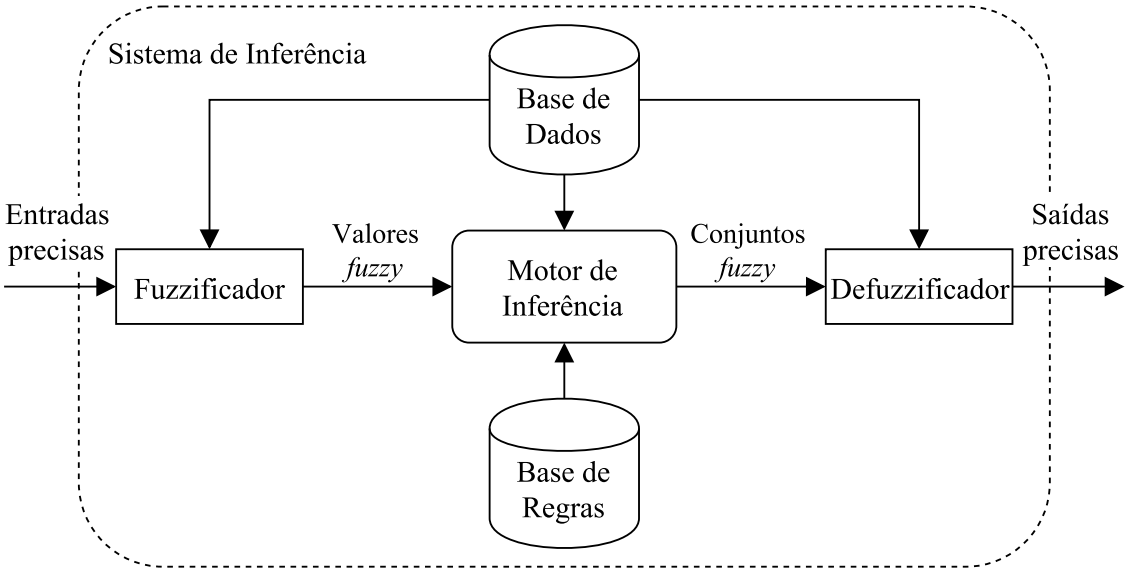


Figura 2 – Componentes de um sistema de inferência *fuzzy*.

Um sistema de inferência *fuzzy* possui três componentes principais (ENGELBRECHT, 2007): (*i*) *Fuzzificador*: responsável por receber as entradas precisas que dão início ao processo de inferência e transformar essas valores para o domínio qualitativo; (*ii*) Motor de inferência: responsável por avaliar as regras do sistema de inferência com os dados de entrada já *fuzzificados* e agregar todos os conjuntos *fuzzy* resultantes de modo a expressar a saída do sistema; (*iii*) *Defuzzificador*: responsável por traduzir a saída do motor de inferência para o domínio quantitativo. As informações necessárias para que o sistema de inferência funcione corretamente estão contidas na sua base de conhecimento, que é constituída pela união da base dos dados sobre as variáveis de entrada e saída do sistema com a base de regras lógicas (*se...então*) que relacionam essas variáveis entre si.

1.5 Trabalhos Relacionados

Esta seção apresenta trabalhos com aspectos relacionados à abordagem proposta. No melhor conhecimento do autor, não foram encontradas pesquisas voltadas especificamente para a manipulação de dados contextuais por meio de bibliotecas no contexto dos jogos pervasivos móveis no momento da escrita desse trabalho. Em uma revisão sistemática realizada por Viana et al. (2014), os trabalhos voltados para utilização de sensores frequentemente constroem jogos pervasivos para elicitar os desafios encontrados no desenvolvimento desses jogos, e apresentam ferramentas de autoria como uma solução. Por esse motivo, as próximas seções discorrem sobre soluções para problemas distintos, mas que compartilham características com a abordagem proposta, nomeadamente, no uso de abstrações contextuais e no tratamento de incertezas.

1.5.1 Abstrações do contexto

Diversas soluções para o tratamento de dados contextuais utilizam arquiteturas baseadas em *middlewares*, ou seja, sistemas que se comportam como mediadores entre uma camada tecnológica anterior e a aplicação. Entre essas soluções, encontra-se o *Context Toolkit* (DEY; ABOWD; SALBER, 2001), um dos primeiros *middlewares* para aplicações sensíveis ao contexto distribuídas, cujo *framework* conceitual foi uma referência da solução proposta nesse trabalho. A abstração utilizada pelo *Context Toolkit* inspira-se nas bibliotecas de desenvolvimento de aplicações gráficas, generalizando o conceito de *Widget* para as aplicações sensíveis ao contexto com o objetivo de separar as responsabilidades de aquisição e interpretação dos dados contextuais da utilização desses por essas aplicações (DEY et al., 1999). No entanto, o *Context Toolkit* não foi projetado para aplicações móveis centralizadas, tornando algumas de suas abstrações e processos desnecessárias ou não aplicáveis na infraestrutura das aplicações móveis. Um exemplo dessa incompatibilidade é o requisito de sempre oferecer dados e interpretações contextuais, independentemente da demanda por esses dados, cuja execução não seria viável por meio de dispositivos móveis com capacidades de armazenamento e processamento limitadas (YÜRÜR et al., 2016).

Em Ferreira, Kostakos e Dey (2015) é apresentado a ferramenta AWARE, uma plataforma para instrumentação de pesquisas com dispositivos móveis, um *framework* para o desenvolvimento de aplicações sensíveis ao contexto em dispositivos móveis, e uma aplicação móvel para facilitar o acesso a dados contextuais. A ferramenta foi construída para solucionar três problemas: (*i*) conduzir experimentos com pessoas utilizando dispositivos móveis é uma tarefa trabalhosa; (*ii*) o desenvolvimento de aplicações sensíveis ao contexto em telefones móveis é tipicamente feito do zero; e (*iii*) usuários finais precisam instalar mais de uma aplicação para tornar seus dispositivos mais sensíveis ao contexto. As principais contribuições do trabalho são uma infraestrutura para suportar pesquisadores na condução de experimentos com dispositivos móveis e uma biblioteca que provê o acesso à essa infraestrutura para desenvolvedores. A API do AWARE foi construída com base no *framework* conceitual do *Context Toolkit* (SALBER; DEY; ABOWD, 1999). O trabalho aponta como limitações a dependência na infraestrutura oferecida pela plataforma móvel e a falta de procedimentos específico para garantir a privacidade dos usuários. Além disso, com um alto número de aplicações de terceiros que dependem da aplicação AWARE cliente, e a necessidade de manter dados contextuais constantemente disponíveis devido pode implicar em problemas futuros relacionados ao uso de bateria e armazenamento. Esses aspectos foram levados em consideração pela solução proposta nesse trabalho.

1.5.2 Tratamento de incertezas

Como os jogos pervasivos móveis dependem de tecnologias de identificação de contexto para prover novas formas de interação (KASAPAKIS; GAVALAS, 2015), as

imperfeições inerentes a essas tecnologias impõem desafios para o processo de desenvolvimento de jogos pervasivos. Na literatura, diversos trabalhos reportam os diferentes tipos de incerteza, seus efeitos no desenvolvimento de jogos pervasivos móveis, como a experiência do jogador pode ser afetada e possíveis soluções.

1.5.2.1 Incertezas em jogos pervasivos

No trabalho de Benford et al. (2003), os autores abordam as dificuldades de desenvolver e orquestrar um jogo pervasivo baseado em localização. No jogo *Can you See me Now?*, desenvolvido pelos autores, a presença de incerteza é uma consequência das tecnologias empregadas pelo jogo, nomeadamente os dispositivos para detecção da localização do jogador. Para solucionar os problemas enfrentados, os autores decidem utilizar artifícios de *design* do jogo para lidar com os dados contextuais incertos, como por exemplo, escondendo as informações incertas do jogador e utilizando filtros para exibir apenas localizações válidas. Essa estratégia corresponde ao item 2 apresentado no estudo de Valente, Feijó e Leite (2017) para o tratamento de incertezas.

No jogo *Mythical: The Mobile Awakening*, desenvolvido por Becam e Nenonen (2008), algumas considerações sobre o desenvolvimento de jogos pervasivos foram discutidas. Uma atenção especial foi dedicada à qualidade das informações contextuais e à confiabilidade dos provedores dessas informações, e foram consideradas as consequências de utilizar dados contextuais diretamente (isto é, sem utilizar filtros ou validar a integridade dos dados) na lógica do jogo. Becam e Nenonen (2008) recomendam evitar o uso de dados contextuais para controlar interações importantes dentro do jogo, uma vez que erros e imprecisões nas leituras dos sensores podem prejudicar a qualidade ou interromper completamente o jogo.

A maioria dos desenvolvedores de jogos pervasivos optam por contornar os problemas causados pelos dados contextuais por meio de manobras criativas relacionadas ao *game design*, como remover a fonte de incerteza diretamente do tema do jogo ou confiando no jogador para informar o contexto desejado, ou incorporar a incerteza como parte do jogo e informar aos jogadores (VALENTE; FEIJÓ; LEITE, 2013). Um exemplo de utilização dessa abordagem foi apresentado em Robert-Bouchard, Dupire e Cubaud (2015 apud CHALMERS et al., 2004, p. 2). No jogo multi-jogador desenvolvido pelos autores, os jogadores são explicitamente alertados sobre a força do sinal de GPS. Essa informação pode ser utilizada para a vantagem do jogador, que pode dirigir-se para áreas com baixa cobertura do GPS com o intuito de surpreender outros jogadores.

1.5.2.2 Incertezas em aplicações sensíveis ao contexto

É importante mencionar que lidar com a incerteza contextual não é um problema exclusivo dos jogos pervasivos, mas sim de todas as aplicações sensíveis ao contexto (BETTINI et al., 2010). As soluções apresentadas nesta seção, embora possam ser modificadas

para suportar a implantação de jogos pervasivos móveis, limitam-se a ambientes físicos fixos e, por isso, não fazem proveito do potencial de mobilidade oferecido pelos *smartphones*.

No trabalho de Ranganathan, Al-Muhtadi e Campbell (2004), foi apresentada uma infraestrutura de suporte a aplicações sensíveis ao contexto, chamada Gaia, capaz de "raciocinar" sob condições de incerteza. O modelo de raciocínio utilizado representa dados contextuais por meio de predicados apresentando graus de confiança. A infraestrutura Gaia aplica esses graus de confiança de diferentes formas, dependendo do mecanismo de raciocínio escolhido. Por exemplo, caso a ferramenta utilize um mecanismo de inferência probabilística e outro de inferência *fuzzy*, um grau de confiança representa a uma probabilidade de um evento ocorrer e um valor verdade de uma proposição *fuzzy*, respectivamente (RANGANATHAN; AL-MUHTADI; CAMPBELL, 2004).

Em Lei et al. (2002), os autores sugerem um serviço para administrar as informações contextuais e decidir quais informações devem ser utilizadas por meio da atribuição de métricas como a qualidade, precisão e relevância para cada informação do contexto. Em outros trabalhos, a sensibilidade ao contexto passa a ser chamada de "sensibilidade a situação", ou seja, uma interpretação de alto nível que classifica dados contextuais e aplica técnicas de reconhecimento de padrões para identificar situações contextuais (ANAGNOSTOPOULOS; HADJIEFTHYMIADES, 2008; BAUMGARTNER et al., 2010).

Por exemplo, no trabalho de Haghighi et al. (2008), os autores combinam conceitos de lógica *fuzzy* com um modelo conceitual para classificar dados contextuais com base em situações pré-estabelecidas. Essa combinação permite o uso de um sistema de inferência *fuzzy* para classificar diversas situações modeladas pela abordagem, e oferecer suporte à tomada de decisões com base nas situações identificadas. Tais trabalhos visam oferecer suporte a aplicações em ambientes ubíquos, isto é, espaços físicos permeados por dispositivos computacionais (BALDAUF; DUSTDAR; ROSENBERG, 2007). A Tabela 1 apresenta uma comparação das ferramentas em relação aos requisitos apresentados na Seção 2.1: separação de responsabilidades (A), geração de dados (B), extensibilidade (C), interpretação do contexto (D), agregação de dados (E) e tratamento de incertezas (F).

Tabela 1 – Comparativo dos requisitos contemplados na literatura.

	A	**B**	**C**	**D**	**E**	**F**
Dey (2001)	✓	✓	-	✓	✓	-
Ferreira (2015)	✓	✓	✓	✓	✓	-
Ranganathan (2004)	✓	✓	-	✓	✓	✓
Lei (2002)	-	✓	-	✓	✓	-
Haghighi (2008)	-	✓	-	✓	✓	✓

2 Solução Proposta

Este capítulo apresenta a abordagem proposta para a manipulação de dados contextuais em jogos pervasivos móveis. Primeiro, são apresentados os requisitos da abordagem que delimitam a especificação de um modelo conceitual, detalhado em seguida. Por fim, é apresentada uma biblioteca que implementa o modelo conceitual especificado. A especificação do modelo e arquitetura da biblioteca são ilustradas ao longo do capítulo pelas visões apresentadas em Kruchten (1995).

2.1 Requisitos

Para produzir a principal contribuição deste trabalho, uma biblioteca para auxiliar o desenvolvedor de JPMs com a manipulação do contexto, é importante determinar quais são as características de uma boa biblioteca (MYERS; STYLOS, 2016). A seguir, são apresentados os requisitos levantados por meio de uma revisão não-sistemática da literatura, especificamente nas áreas: arquitetura de *software*, *design* de APIs e aplicações móveis cientes do contexto.

A. Separação de responsabilidades: Os processos de aquisição e utilização dos dados contextuais devem ser desenvolvidos de forma independente, permitindo uma maior flexibilidade, facilitando o reúso e reduzindo acoplamento (DEY; ABOWD; SALBER, 2001; HENNING, 2009; MAIA et al., 2013; FERREIRA; KOSTAKOS; DEY, 2015).

B. Geração de dados: O mecanismo de aquisição dos dados contextuais deve ser flexível o suficiente para permitir a utilização de dados gerados diretamente pelos jogadores (FERREIRA; KOSTAKOS; DEY, 2015; KASAPAKIS; GAVALAS, 2016).

C. Extensibilidade: Uma biblioteca de desenvolvimento orientada a objetos precisa facilitar a extensão de funcionalidades; os mecanismos de aquisição e interpretação dos dados contextuais precisam dar suporte a comportamentos personalizados (BLANCHETTE, 2008; HENNING, 2009; FERREIRA; KOSTAKOS; DEY, 2015; BLOCH, 2006).

D. Interpretação do contexto: Os mecanismos de interpretação dos dados contextuais devem permitir que novas informações de alto nível sejam produzidas a partir de outros dados em níveis de abstração mais baixos (DEY; ABOWD; SALBER, 2001; MAIA et al., 2013; FERREIRA; KOSTAKOS; DEY, 2015; YÜRÜR et al., 2016).

E. Agregação de dados: Os mecanismos de interpretação precisam permitir a agregação dos dados contextuais relacionados (DEY; ABOWD; SALBER, 2001; MAIA et al., 2013; FERREIRA; KOSTAKOS; DEY, 2015).

F. Tratamento de incertezas: Os mecanismos de interpretação devem ser capazes de lidar com dados contextuais dotados de imprecisões e incertezas (BALDAUF; DUSTDAR; ROSENBERG, 2007; YÜRÜR et al., 2016).

Além desses requisitos, também foram levadas em consideração um conjunto de boas práticas para o desenvolvimento de APIs na implementação da biblioteca para a linguagem Kotlin, que são mencionadas na Seção 2.3.

2.2 Modelo Conceitual

A partir dos requisitos definidos, foi concebido um modelo conceitual para representar entradas contextuais e estabelecer um processo de manipulação contextual. O papel do modelo é orientar a implementação de uma biblioteca para manipulação de dados contextuais nos JPMs. O modelo é baseado em uma abstração orientada a objetos que define quatro entidades básicas. A abstração empregada e o papel de cada entidade no processo de manipulação são apresentados na subseção a seguir. Em seguida, o processo de manipulação do contexto é exemplificado como um todo.

2.2.1 Abstração do domínio

Em um jogo digital, o jogador age no universo do jogo por meio da interação física com os dispositivos de entrada, como um controle *video-game*, suportados pela plataforma. O jogador pode interagir com o controle de inúmeras formas, mas somente aquelas interações que são observadas, como um botão, irão incitar o controle a gerar um sinal digital que poderá ser interpretado pelo jogo. Nos jogos pervasivos, o contexto do jogador também faz parte de sua entrada e, metaforicamente, ele pode ser visto como um jogador. Com base nessa reflexão, foram identificados quatro elementos que, quando utilizados em conjunto, compõem uma abstração do processo de manipulação dos dados contextuais descrito: o ator ou *fonte* dos sinais de entrada; os *botões* que são ativados por esses sinais; a unidade que *interpreta* se um dado botão deve ser ativado; e o *controle* que agrupa botões relacionados. Estes elementos são representados respectivamente pelas entidades `Fonte`, `Entrada`, `Interpretador` e `Controle`, apresentadas pela Figura 3.

2.2.1.1 Fonte

As *fontes* dos sinais de entrada, representadas pela classe abstrata `Fonte` (ver Figura 4), são os pontos de origem dos dados contextuais no modelo proposto. Subclasses

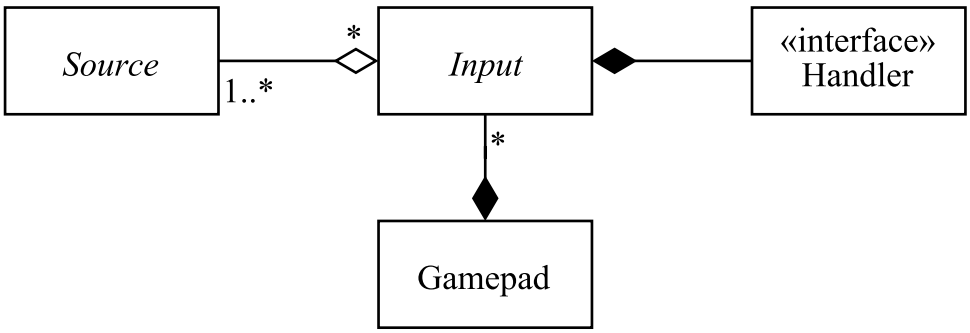


Figura 3 – Visão lógica – Relacionamentos entre as entidades do modelo.

desta classe abstrata são responsáveis por adquirir e distribuir quaisquer dados que possam ser considerados úteis para os jogos pervasivos móveis, contemplando o requisito A (separação de responsabilidades). Tipicamente, estes são dados detectados pelos sensores embutidos dos dispositivos móveis.

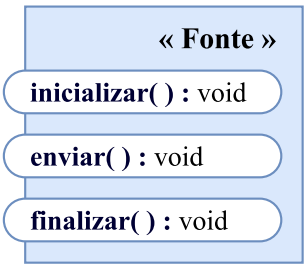


Figura 4 – Interface do tipo abstrato `Fonte`.

Embora as plataformas móveis modernas abstraiam o uso dos sensores para as aplicações — pois suas infraestruturas oferecem uma camada de abstração do *hardware* (*Hardware Abstraction Layer*, HAL) — o desenvolvedor precisa conhecer os diferentes sensores em uso antes de utilizar e/ou combinar os dados sensoreados. Além disso, a definição do contexto abrange diferentes tipos de dados (DEY, 2001), de modo que os valores sensoreados sejam apenas um tipo de dado contextual. Logo, o papel da entidade `Fonte` é proporcionar uma interface homogênea para abstrair as diferentes formas de aquisição e preprocessamento dos dados contextuais. Essa característica está diretamente relacionada com os requisitos B e D (geração de dados e interpretação do contexto).

2.2.1.2 Botão

O propósito de um *botão* ativador, representado pela entidade `Entrada` do modelo proposto (ver Figura 5), é definir como o jogo pervasivo móvel interpreta os dados de uma ou mais subclasses de `Fonte`, e abstrair como esses dados são transformados em um evento de entrada, que virá a ser processado pelo jogo. Por esse motivo, parte do comportamento de um objeto `Entrada`, assim como o seu conjunto de objetos `Fonte` utilizados, só podem

ser determinados no momento da sua instanciação, quando as regras do jogo são conhecidas pelo desenvolvedor.

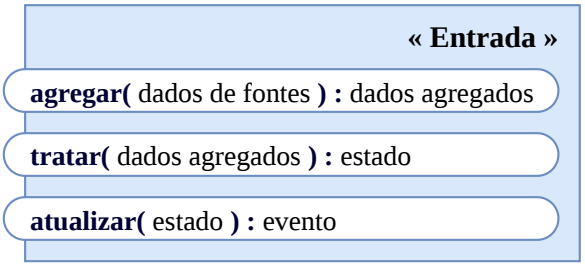


Figura 5 – Interface do tipo abstrato `Entrada`.

Este botão ativador pode ser visto como uma máquina de estados. Logo, uma subclasse de `Entrada` precisa determinar o tipo de dado que representa o seu estado interno. No momento que os dados das fontes são recebidos pelo botão, ele pode determinar se uma transição de estado será necessária e se um evento de entrada será gerado. Devido a essas características, o desenvolvedor do jogo pervasivo pode agregar e interpretar dados contextuais, contemplando os requisitos D e E (interpretação e agregação de dados).

2.2.1.3 Interpretador

Os *interpretadores dos dados do contexto* são objetos que representam um mapeamento de um conjunto de valores de entrada para um conjunto de valores de saída (ver Figura 6). O objetivo da entidade `Interpretador` é abstrair o processo de manipulação dos dados contextuais inicializado pelos botões; qualquer função que mapeie entradas em saídas pode ser implementada como um `Interpretador`, facilitando sua reutilização. Com isso, é possível aplicar algoritmos de preprocessamento de sinais para lidar com incertezas nos dados contextuais, cumprindo o requisito F (tratamento de incertezas).

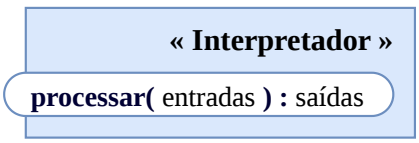


Figura 6 – Interface `Interpretador`.

2.2.1.4 Controle

O *controle* agrupador de botões ativadores é o último componente do modelo conceitual. O papel de um `Controle` é agir como um recipiente de instâncias da entidade `Entrada`, ou seja, ele agrupa todas as instâncias de botões que estejam relacionadas a um jogo específico. Além disso, o `Controle` é responsável por proporcionar um meio de comunicação entre os eventos dos botões e o *loop* do jogo. Logo, a implementação de um `Controle` pode assumir diferentes formas e dependerá da plataforma alvo, especificamente

do sistema de processamento de eventos empregado pela API dessa plataforma (ver Figura 7).

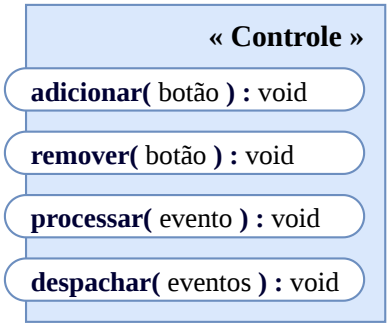


Figura 7 – Interface da entidade `Controle`.

2.2.2 Manipulação do contexto

Por meio das quatro entidades definidas, é possível executar o processo de manipulação do contexto de forma modular. Na perspectiva dos dados contextuais, esse processo é realizado em três etapas: aquisição, interpretação e utilização. As subclasses de `Fonte` são exclusivamente responsáveis pela aquisição. Porém, estes objetos também podem interpretar os dados, elevando o nível de abstração. As subclasses de `Entrada` interpretam os dados de um ou mais objetos `Fonte` e geram eventos. Instâncias de `Interpretador` podem ser implementadas para guardar procedimentos de interpretação reutilizáveis. Por fim, um `Controle` agrupa objetos `Entrada` relacionados e despacha seus eventos para o jogo. Esse processo de manipulação do contexto é ilustrado na Figura 8. As próximas seções detalham o fluxo dos dados em cada etapa.

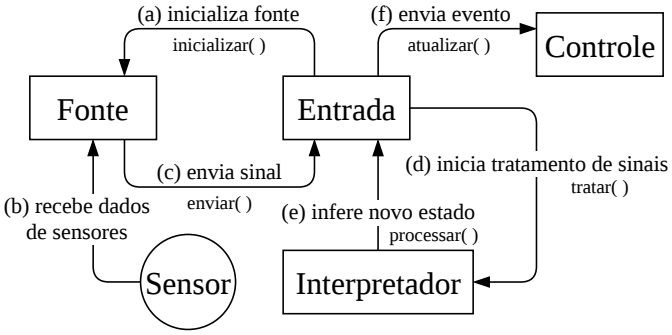


Figura 8 – Processo de manipulação dos dados contextuais.

2.2.2.1 Aquisição dos dados contextuais

Nessa etapa, as fontes obtêm os dados contextuais que resultam nos sinais de entrada esperados pelos botões. Esse processo está estritamente relacionado com o ciclo de vida da entidade `Fonte`. Antes que uma fonte possa enviar sinais, é necessário que ela esteja ativa, ou seja, a fonte precisa ser requisitada por pelo menos um `Entrada` (a). Esse

procedimento de ativação corresponde a uma chamada ao método `inicializar`, de modo a realizar todas as preparações para que a `Fonte` possa enviar sinais, por exemplo, como configurações de sensores. A cada novo sinal recebido (b), o método `enviar` é chamado para despachar os sinais aos botões interessados (c).

2.2.2.2 Interpretação do contexto

Os dados contextuais obtidos pelas fontes podem ser interpretados em três momentos diferentes. A própria subclasse de `Fonte`, responsável pela aquisição, pode elevar o nível de abstração dos dados contextuais antes de despachá-los para um `Entrada`. Como uma `Fonte` pode oferecer dados para mais de um `Entrada` e pode ser utilizada por jogos diferentes, interpretações de dados realizadas em uma `Fonte` não são recomendadas. Basicamente, existe uma troca entre especificidade–generalidade; uma `Fonte` genérica pode ser reutilizada em diferentes jogos, porém uma `Fonte` específica pode facilitar a implementação de um jogo.

A interpretação dos dados, com base nas especificidades do jogo, é geralmente realizada pelas subclasses de `Entrada`. O estado atual de um botão sempre é recalculado quando novos sinais são recebidos de acordo com o método `tratar` (d). A implementação desse procedimento pode ser abstraída por meio de um `Interpretador` (e). Após determinar o novo estado, o `Entrada` envia um evento para o objeto `Controle` (f).

2.2.2.3 Utilização do contexto

Para que os eventos produzidos pelos botões sejam processados pelo *loop* do jogo, o objeto `Controle` faz o papel de intermediador, traduzindo os eventos dos botões para um formato adequado para a plataforma alvo. Após realizada essa tradução, os eventos podem ser utilizados pelo jogo da mesma maneira que os outros eventos de entrada convencionais. Essa etapa marca o fim do processo de manipulação dos dados contextuais orquestrado pelas entidades do modelo proposto, e o início da lógica do jogo.

2.3 Biblioteca Gamepad

Esta seção apresenta a implementação de uma biblioteca, denominada Gamepad, baseada no modelo conceitual especificado. Durante o desenvolvimento dessa biblioteca, buscou-se aplicar boas práticas no *design* da API para melhorar a usabilidade da solução (BLOCH, 2006). A biblioteca foi desenvolvida na linguagem Kotlin[1], com o intuito de suportar, inicialmente, jogos pervasivos para a plataforma Android (BUTLER, 2011; JETBRAINS, 2018). Entretanto, é válido mencionar que a generalidade da especificação do modelo conceitual possibilita a implementação da biblioteca em outras plataformas.

[1] Disponível em: <https://kotlinlang.org/>

As classes implementadas pela biblioteca Gamepad são apresentadas na Figura 9. As entidades `Fonte`, `Entrada`, `Interpretador` e `Controle` do modelo conceitual foram implementadas na biblioteca por meio das classes de mesmo nome. A super-classe `Context Element` foi adicionada para esconder particularidades da plataforma Android. Os demais objetos são apresentados nas subseções seguintes.

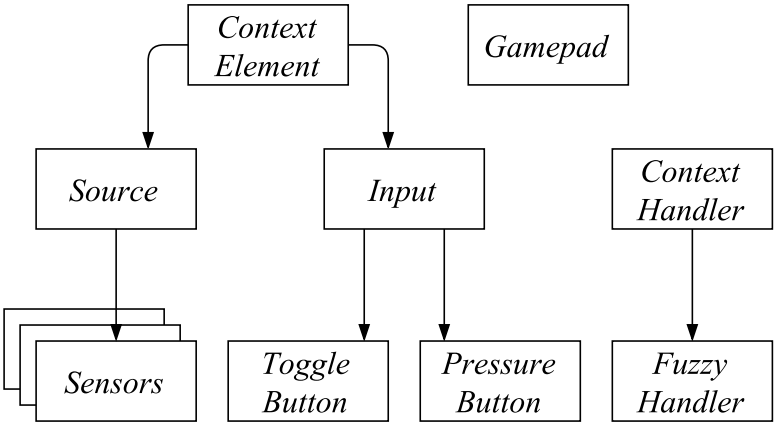


Figura 9 – Hierarquia dos objetos da biblioteca.

2.3.1 Implementações

Para o desenvolvedor do jogo pervasivo, o uso da biblioteca pode ser dividido em três etapas: (*i*) implementar `Fonte` e `Entrada` que apresentem comportamento desejável; (*ii*) instanciar e configurar os botões de acordo com as regras do jogo; (*iii*) implementar a lógica do jogo com base nos eventos dos novos botões. Dessa forma, é possível classificar o processo de desenvolvimento em quatro camadas (ver Figura 10), na qual cada camada pode ser desenvolvida de maneira independente. A biblioteca ainda oferece um conjunto de implementações prontas para uso, tornando desnecessária a primeira etapa. Contudo, caso seja necessário implementar um comportamento específico, é possível estender as classes abstratas `Fonte` e `Entrada`, contemplando o requisito C (extensibilidade).

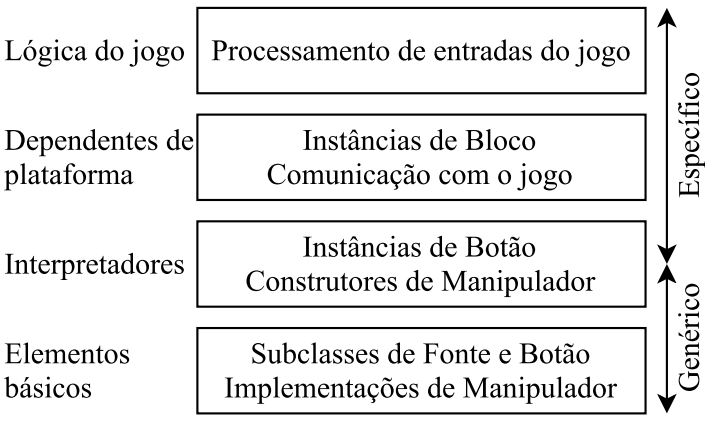


Figura 10 – Visão em 4 camadas da arquitetura de desenvolvimento.

2.3.1.1 Fontes

Os sensores embutidos da plataforma Android são claros exemplos de `Fonte` no modelo conceitual. Como o acesso aos sensores é disponibilizado por meio de interfaces do SDK (*Software Development Kit*) da plataforma, é possível mapear todos os sensores para fontes seguindo o mesmo padrão de implementação apresentado pela Listagem 1.

```
1  class SensorImpl(context: Context) : Source<FloatArray>(context) {
2    override fun onBegin() { // Registrar um 'ouvinte' do sensor.
3      manager.registerListener(ouvinte, sensor, SENSOR_DELAY_NORMAL)
4    }
5    override fun onEnd() { // Remover o 'ouvinte' do registro.
6      manager.unregisterListener(ouvinte)
7    }
8    /** Dentro do 'ouvinte' **/
9    override fun onSensorChanged(event: SensorEvent) { // Atualizar.
10     ultimosDados = event.values
11     onData()
12   }
13   /** Fora do 'ouvinte' **/
14   override fun onData() { // Enviar últimos dados.
15     broadcast(ultimosDados)
16   }
17 }
```

Listagem 1 – Exemplo genérico de `Fonte` baseada em sensor Android.

2.3.1.2 Entradas

Na implementação atual da biblioteca, existem dois tipos básicos de botões: `ToggleButton` e `PressureButton`. A principal diferença entre os dois botões está no tipo de dado utilizado para representar o estado interno de cada botão. Enquanto o `ToggleButton` equivale a um botão simples com apenas dois estados, pressionado e solto, o `PressureButton` é um botão capaz de perceber a intensidade, ou pressão, do "toque" e o seu estado corresponde a um valor dentro do intervalo $[0, 1]$.

2.3.1.3 Interpretadores

Como foi mencionado na Subseção 2.2.2.2, a interface `Interpretador` abstrai o processo de interpretação dos dados contextuais nos botões e permite que qualquer função ou mapeamento de entradas em saídas seja implementado como um manipulador. A atual implementação da biblioteca oferece um único manipulador, `FuzzyHandler`, que realiza um processo de inferência *fuzzy*. Os parâmetros do motor de inferência *fuzzy* são obtidos por meio de uma função construtora (ver Listagem 2).

2.3.2 Utilização da biblioteca

Sob a perspectiva do desenvolvedor, os jogos pervasivos construídos com a biblioteca Gamepad são semelhantes aos jogos digitais convencionais. Os dados contextuais passam

```
fun<T, U> fuzzyHandler(
  inputs   : List<FuzzyInput>, /* Parâmetros do sistema fuzzy */
  outputs  : List<FuzzyOutput>,
  rulebase : List<FuzzyRule>,
  operators: List<FuzzyOperator>,
  before   : ( Pair<T, U> ) -> Array<Float>, /* Transformações */
  after    : (Array<Float>) -> T
) : (Pair<T, U>) -> T /* Lambda 'handler' resultante */
```

Listagem 2 – Assinatura da função construtora do `FuzzyHandler`.

por um mapeamento realizado por instâncias da classe `Entrada` para que possam ser transmitidos ao *loop* do jogo na forma de eventos. Dessa forma, quanto melhores forem as abstrações aplicadas aos dados contextuais, mais simples torna-se o processamento dos eventos de entrada dentro do *loop* do jogo.

No entanto, nem sempre uma informação pertencente ao contexto do jogador precisará passar por interpretações complexas. Um exemplo comum são os dados obtidos pelo GPS, um par de números representando a latitude e longitude do usuário, que podem ser utilizados diretamente por componentes visuais da plataforma, como um mapa. Em casos semelhantes a esse, a biblioteca Gamepad ainda pode ser utilizada apenas como um meio para transmitir esses dados até o jogo sem a necessidade de interpretá-los. A subseção seguinte exemplifica a utilização da biblioteca em um jogo pervasivo de cartas.

2.3.2.1 Jogo pervasivo de cartas

A ideia principal em um jogo de cartas é utilizar informações sobre o contexto do jogador para afetar os atributos dessas cartas durante interações no jogo. Por exemplo, em um jogo de batalhas utilizando cartas com atributos que indiquem sua força, é possível incrementar ou diminuir a força de uma carta dependendo da temperatura ambiente atual. Por meio da biblioteca proposta, esse jogo pode ser implementado com uma única `Fonte` de dados e dois botões do tipo `PressureButton` utilizando um sistema de inferência *fuzzy* (*Fuzzy Inference System*, FIS), implementado por um `Interpretador` contextual, como ilustrado na Figura 11.

A fonte dos dados (o objeto `Fonte`) seria responsável por capturar a temperatura no ambiente do jogador, informações que podem ser obtidas por meio de APIs ou sensores embutidos em dispositivos modernos. Em seguida, as instâncias dos botões (`Entrada`) devem ser associadas com a fonte de temperatura e cada instância deve utilizar um manipulador *fuzzy* (`Interpretador`). Os dois manipuladores seriam responsáveis por *fuzzificar* a temperatura e inferir se a mesma é *fria* ou *quente*. Por fim, os botões enviam um valor *defuzzificado* dentro do intervalo $[0, 1]$ para o jogo, representando a intensidade da temperatura atual nos conceitos *frio* e *quente*. Dessa forma, o desenvolvedor pode fazer uso desses valores, representativos da intensidade da temperatura, para definir o poder

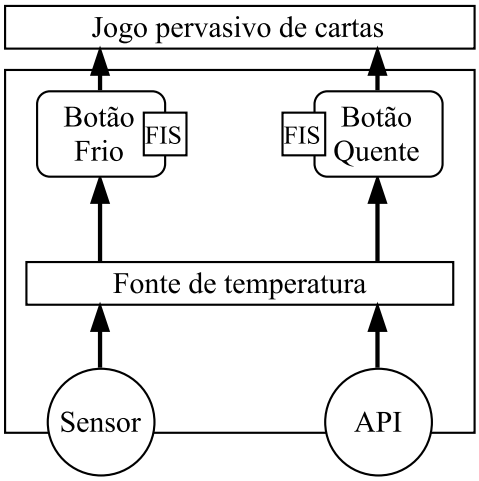


Figura 11 – Arquitetura do jogo pervasivo de cartas.

de uma carta levando em consideração o significado linguístico dessas variáveis que foi proporcionado pela modelagem *fuzzy*.

2.4 Considerações Finais

Neste capítulo foram apresentados os elementos da abordagem proposta para a representação e manipulação dos dados contextuais em jogos pervasivos móveis. A abordagem consiste em uma biblioteca de desenvolvimento para estruturar os dados do contexto. A construção do modelo foi realizada a partir de um conjunto de requisitos levantados na literatura. O modelo conceitual utilizou uma abstração dos sistemas de processamento de entradas em jogos digitais.

A biblioteca, construída com base no modelo conceitual, implementa as entidades especificadas para realizar a manipulação dos dados contextuais nos JPMs de forma modular, contemplando o requisito A, separação de responsabilidades. Por meio das entidades `Fonte` e `Entrada`, é possível implementar os procedimentos de aquisição e interpretação dos dados contextuais, contemplando os requisitos B e D, geração de dados e interpretação do contexto. A entidade `Entrada` também realiza um papel de agregador, podendo associar informações de diferentes objetos `Fonte`, contemplando o requisito E, agregação de dados. A interface `Interpretador` permite a generalização de qualquer técnica de preprocessamento, contemplando o requisito F, tratamento de incertezas. Por fim, o comportamento interno de cada um destes componentes foi protegido de alterações que fujam do fluxo de controle padrão e o comportamento específico das entidades foi tornado público para que seja possível estendê-las, contemplando o requisito C, extensibilidade.

3 Prova de Conceito

Este capítulo apresenta uma prova de conceito para validar a aplicabilidade da biblioteca proposta. Foi desenvolvido um jogo pervasivo móvel chamado “Radar”. O jogo faz parte do gênero de jogos de caça-ao-tesouro, cujas regras são descritas como segue: o tesouro e jogador são as entidades fundamentais no jogo; os tesouros são objetos espalhados em um mapa virtual que corresponde a um ambiente físico; os jogadores precisam se movimentar para atualizar sua posição no jogo e encontrar todos os tesouros. Em geral, jogos de caça-ao-tesouro auxiliam os jogadores a encontrar os tesouros por meio de dicas ou mapas exibidos na tela do *smartphone* (ECONOMOU et al., 2015). Com base nessas linhas gerais, as próximas seções apresentam o processo de design e implementação do jogo utilizando a biblioteca proposta.

3.1 Descrição do Jogo

Radar é um jogo pervasivo em tempo real para um jogador em ambientes abertos, onde o jogador recebe a missão de encontrar um conjunto de tesouros projetados nas localidades ao seu redor. A fantasia do jogo consiste em imaginar o *smartphone* do jogador como um “radar”, um dispositivo capaz de exibir a localização dos tesouros em relação ao jogador, porém, esse dispositivo apresenta uma falha que prejudica a sua capacidade de exibir as localizações na medida que o jogador se move. Especificamente, o radar é afetado por dois dados contextuais: a distância entre o jogador e os tesouros; e o ângulo da tela do *smartphone* em relação aos tesouros; tesouros são “descobertos” individualmente pelo radar em uma frequência variável que depende desses dados; essa frequência é maior quando o jogador está de frente com um tesouro e quando o jogador está próximo ao tesouro. Por exemplo, um tesouro em frente ao jogador será descoberto com maior frequência do que um tesouro distante e atrás do jogador. Essas mecânicas de jogo exemplificam como dados contextuais podem ser utilizados para enriquecer a experiência de jogos móveis.

Apesar de que a descrição de mecânicas baseadas em dados contextuais seja simples e direta, existem diversos desafios do ponto de vista da implementação. Devido à imprecisão inerente aos sensores, é preciso ter cuidado antes de utilizar dados contextuais para controlar interações importantes do jogo (BECAM; NENONEN, 2008), para evitar que possíveis incertezas prejudiquem a experiência do jogador. Por exemplo, se o jogador precisa permanecer a 5 metros de distância de um tesouro por 1 segundo para capturá-lo, mesmo que o jogador não se mova, uma única atualização de localização imprecisa pode informar que o jogador se moveu para fora do limite e interromper a captura. Esse tipo de situação pode ser frustrante para os jogadores e precisa ser evitado. Por esse motivo, a

biblioteca proposta expõe para o desenvolvedor janelas para o tratamento de incertezas dos dados contextuais. Utilizando conjuntos *fuzzy*, a interação mencionada anteriormente poderia verificar se o jogador estivesse *perto* do tesouro por *pouco* tempo, na qual os termos linguísticos associados aos conjuntos *perto* e *pouco* levam em consideração as variabilidades e incertezas desses valores sensoreados.

3.2 Implementação

A partir da descrição das mecânicas do jogo por meio de termos linguísticos, é possível identificar e compreender o relacionamento entre as entradas e saídas do jogo de forma intuitiva. Porém, geralmente informações de alto-nível precisam ser derivadas de vários tipos de dados contextuais. No caso do jogo Radar, a proximidade entre o jogador e os tesouros, e a direção do jogador em relação ao tesouro são exemplos de informações de alto-nível que precisam ser obtidas. Dessa forma, as próximas subseções descrevem como as entidades da biblioteca foram utilizadas para gerar essas informações.

3.2.1 Aquisição

A primeira informação obtida é o conjunto das distâncias entre o jogador e os tesouros restantes (P), dado pela expressão $P = \{d(p, t_i) \mid i \leq N\}$, na qual N é a quantidade de tesouros restantes, t_i representa a posição do i-ésimo tesouro, p corresponde à posição do jogador e $d(x, y)$ uma função que retorna a distância euclidiana em metros entre as posições de x e y. Cada posição é representada por um par (lat, lon), com os valores de latitude e longitude da entidade. A posição do jogador (p) é detectada com a ajuda do GPS, enquanto as posições dos tesouros são geradas arbitrariamente pelo jogo.

Para adquirir a diferença angular entre o "rosto" do jogador e cada um dos N tesouros restante (Θ), os sensores de campos magnéticos (magnetômetro) e de aceleração linear nos eixos do *smartphone* (acelerômetro) são utilizados para estimar a direção apontada pelo dispositivo móvel (θ^p) em graus relativos ao norte verdadeiro. Esses tipos de sensores estão presentes na maioria dos dispositivos Android, e são frequentemente utilizados na literatura (ZIEGLER et al., 2009; MUKHOPADHYAY, 2015). Em seguida, é possível calcular o ângulo de rolamento inicial do tesouro i para o jogador (θ_i^t) em graus relativos ao norte verdadeiro por meio dos serviços de localização oferecidos pela plataforma Android. Dessa forma, o conjunto Θ pode ser obtido pela expressão $\Theta = \{f(\theta^p, \theta_i^t) \mid i \leq N\}$, na qual $f(a, b)$ é uma função que retorna a menor diferença entre os ângulos a e b, dentro do intervalo $[0, 180]$, em graus.

Como a quantidade de tesouros restantes (N) é um valor decrementado sempre que um tesouro for encontrado, é necessário implementar uma subclasse da entidade `Fonte` capaz de receber atualizações sobre o estado do jogo. A assinatura da subclasse

é apresentada pela Listagem 3, acompanhada da implementação de três elementos: a propriedade `defaultData`, que inicializa uma instância nula do mapa de dados genérico produzido pela fonte; o método `transform`, que converte os dados produzidos para um tipo de dado específico; e duas constantes que são utilizadas como chaves do mapa de dados genérico.

```
1  class CustomSource private constructor(context: Context) : Source<GameData>(context) {
2    companion object: SingletonHolder<CustomSource, Context>(::CustomSource) {
3      val DISTANCES: String = "chave-distancias"
4      val DIFFERENCES: String = "chave-diferencas"
5    }
6    override val defaultData: ContextData by lazy {
7      ContextData(System.currentTimeMillis(),
8        mapOf(DISTANCES to FloatArray(0), DIFFERENCES to FloatArray(0)))
9    }
10   override fun transform(data: ContextData): GameData {
11     val distances = data.values[DISTANCES]!!
12     val differences = data.values[DIFFERENCES]!!
13     return GameData(distances, differences, data.timestamp)
14   }
15 }
```

Listagem 3 – Implementação da subclasse de `Fonte`.

O tipo de dado específico retornado pelo método `transform` da subclasse é utilizado como uma tupla, ou seja, ele tem o propósito de guardar os resultados em um formato estático e simplificar o acesso por parte dos desenvolvedores que consumirão esses dados com objetos `Entrada`. Este tipo de dado, chamado de `GameData`, é apresentado na Listagem 4.

```
1  data class GameData(val dists: FloatArray, val diffs: FloatArray, val timestamp: Long)
```

Listagem 4 – Tipo de dado para guardar os resultados da subclasse de `Fonte`.

Por fim, as posições dos tesouros restantes em conjunto com a posição do jogador são utilizadas pela subclasse para calcular o conjunto das distâncias (P) e dos ângulos de diferença (Θ). Um exemplo da implementação desse procedimento é encontrado na Listagem 5, assumindo que as variáveis `heading`, `pl`, `ts`, `tresuresLeft` e `treasures` representam respectivamente: a diferença angular entre o "rosto" do jogador e o norte magnético; a localização do jogador; o momento do evento mais recente; a quantidade de tesouros restantes; e um vetor com as localizações dos tesouros.

3.2.2 Interpretação

Após a escolha do objeto `Fonte`, o passo seguinte é determinar o tipo de `Entrada` que será utilizado e de qual forma os dados contextuais serão interpretados por seu `Interpretador`. Considerando que os dados retornados pelo `Fonte` escolhido são vetoriais, com cada valor referente ao i-ésimo tesouro, decidiu-se implementar uma versão vetorial

```
fun calculate() {
  val dts = FloatArray(treasuresLeft)
  val dfs = FloatArray(treasuresLeft)
  treasures.forEachIndexed { i, tr ->
    val r = FloatArray(2)
    Location.distanceBetween(pl.latitude, pl.longitude, tr.latitude, tr.longitude, r)
    dts[i] = r[0]
    val diff = r[1] - heading
    dfs[i] = if (diff > 180) diff - 360 else if (diff < -180) diff + 360 else diff
  }
  channel.broadcast(ContextData(ts, mapOf(DISTANCES to dts, DIFFERENCES to dfs)))
}
```

Listagem 5 – Procedimento para calcular e enviar os valores de P e Θ.

do objeto `PressureButton`. O estado desse `Entrada` é um vetor numérico de tamanho N, no qual cada valor representa a *intensidade* que o botão está pressionado para o i-ésimo tesouro. A implementação desse objeto é exemplificada pela Listagem 6.

```
class CustomInput(context: Context) : Input<FloatArray>(context) {
  override var state: FloatArray = FloatArray(0)
  override var eventMapper: Mapping<FloatArray, GamepadEvent> = {
    GamepadEvent(GamepadEvent.ACTION_AXIS, code, axis = it)
  }
  override fun intercept(nextState: FloatArray): Boolean = Arrays.equals(state,
  ↪   nextState)
}
```

Listagem 6 – Implementação da subclasse de `Entrada`.

Em consequência, um objeto `Interpretador` utilizado por esse botão precisa inferir um vetor de tamanho N. Para isso, foi utilizada uma variação da função construtora para o motor de inferência *fuzzy* apresentado na Subseção 2.3.1.3, que encarrega-se de executar o procedimento de inferência para cada valor do vetor de entrada, e de retornar um vetor com todos os resultados de saída na mesma ordem. A implementação dessa função é exemplificada pela Listagem 7.

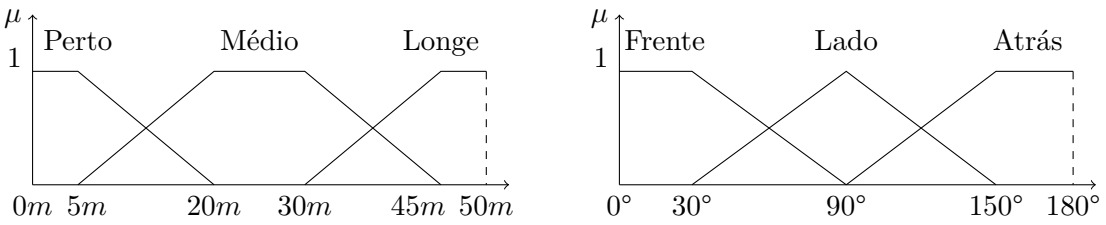
Para representar as distâncias do conjunto P, foram definidos três termos linguísticos *fuzzy*: *Perto*, *Médio* e *Longe*. Já para os ângulos do conjunto Θ, foram definidos os termos: *Frente*, *Lado* e *Atrás*. As funções de pertinência desses termos linguísticos, voltadas para os universos de discurso das entradas P e Θ, são ilustradas pelas Figuras 12a e 12b respectivamente.

Em seguida, é necessário definir a saída do sistema *fuzzy*, que corresponde às intensidades do botão para cada tesouro, de acordo com a mecânica de jogo definida na Seção 3.1. Para representar os valores das intensidades do botão em relação aos tesouros (I), foram definidos os termos linguísticos *fuzzy*: *Fraca*, *Baixa*, *Regular*, *Alta* e *Forte*. A Figura 13 apresenta as funções de pertinência da saída para os valores do conjunto I.

Com a definição das variáveis *fuzzy*, a base de regras utilizada para inferir a

```
fun<T> customHandler(init: Engine.() -> Unit, before: (ContextMap) -> List<DoubleArray>,
    after: (List<DoubleArray>) -> T): (ContextMap) -> T {
  val engine = Engine().apply(init)
  return fun(sensorData: ContextMap): T {
    val inputList = before(sensorData)
    val outputList = mutableListOf<DoubleArray>()
    assert(inputList.size == engine.numberOfInputVariables())
    for (pair in 0 until inputList.first().size) {
      for (item in inputList.withIndex()) {
        engine.getInputVariable(item.index).value = item.value[pair]
      }
      engine.process()
      val outputArray = DoubleArray(engine.numberOfOutputVariables(), {
    engine.getOutputVariable(it).value })
      outputList = outputArray
    }
    return after(outputList)
  }
}
```

Listagem 7 – Implementação da função `Interpretador`.



(a) Proximidade (b) Direção

Figura 12 – Funções de pertinência das entradas (a) P e (b) Θ.

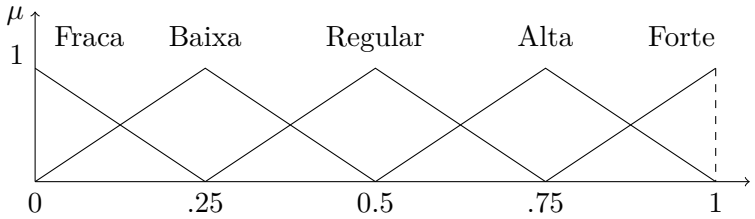


Figura 13 – Funções de pertinência da saída I.

intensidade do tesouro i (I_i) com base na proximidade (p) e no ângulo de diferença (θ) do jogador em relação ao i-ésimo tesouro é apresentada pela Tabela 2.

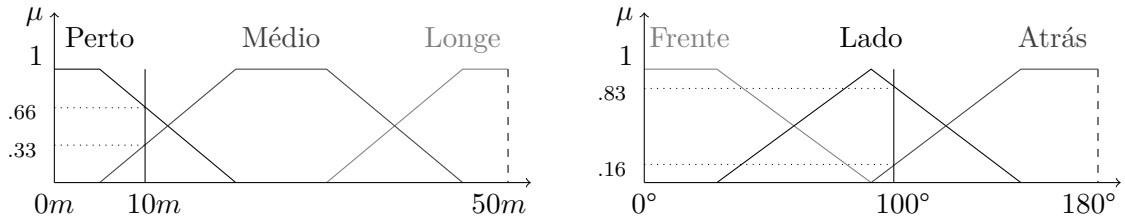
A calibragem do conhecimento deste sistema de inferência *fuzzy*, expressada pela base de regras e pelas funções de pertinência escolhidas, foi realizada de maneira completamente empírica, por meio testes repetitivos com parâmetros distintos e a verificação de seus efeitos na animação do jogo. Foram testadas apenas funções de pertinência simples, com formato trapezoidal e triangular, de modo a simplificar a implementação.

Tabela 2 – Base de regras para inferir as intensidades do botão.

#		p		θ		I_i
1	SE	Perto	E	Frente	ENTÃO	Forte
2	SE	Perto	E	Lado	ENTÃO	Alta
3	SE	Perto	E	Atrás	ENTÃO	Regular
4	SE	Médio	E	Frente	ENTÃO	Alta
5	SE	Médio	E	Lado	ENTÃO	Regular
6	SE	Médio	E	Atrás	ENTÃO	Baixa
7	SE	Longe	E	Frente	ENTÃO	Regular
8	SE	Longe	E	Lado	ENTÃO	Baixa
9	SE	Longe	E	Atrás	ENTÃO	Fraca

3.3 Processo de Inferência

Para exemplificar o processo de inferência realizado pelo motor de inferência *fuzzy* implementado como `Interpretador`, são dados os valores $10m$ e 100° como resultados das respectivas fontes (`Fonte`) de proximidade e direção para um tesouro i. Os resultados do processo de *fuzzificação* são: $p = \{\ Perto(0.66),\ Medio(0.33)\ \}$ e $\theta = \{\ Lado(0.83),\ Atras(0.16)\ \}$. A Figura 14 ilustra o procedimento de *fuzzificação* em relação aos valores mencionados.



(a) (b)

Figura 14 – Processo de *fuzzificação* dos valores (a) $10m$ e (b) 100°

Com base nos graus de pertinência resultantes para os valores de entrada exemplificados anteriormente, a Tabela 3 apresenta as regras ativadas.

Tabela 3 – Regras ativadas pelos valores $10m$ e 100°.

#		p		θ		I_i
2	SE	Perto	E	Lado	ENTÃO	Alto
3	SE	Perto	E	Atrás	ENTÃO	Regular
5	SE	Médio	E	Lado	ENTÃO	Regular
6	SE	Médio	E	Atrás	ENTÃO	Baixo

A ativação da regra 2 é apresentada na Figura 15a e a Figura 15b ilustra o conjunto *fuzzy* resultante da agregação dos conjuntos inferidos. Como resultado, uma intensidade

de valor 0.6 é computada utilizando o método de *defuzzificação* Centro de Gravidade (BROEKHOVEN; BAETS, 2006).

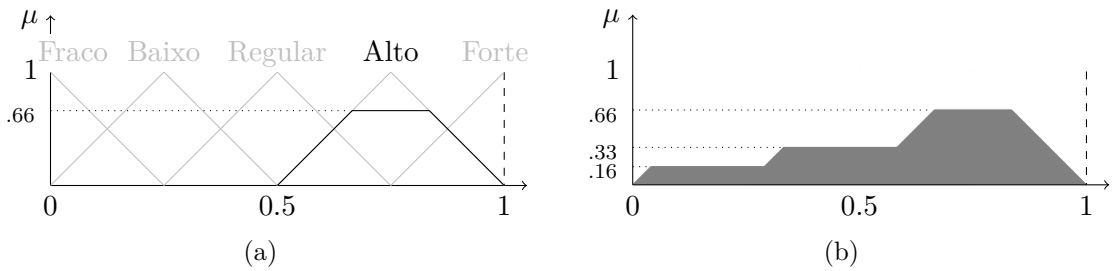


Figura 15 – (a) Ativação da regra 2 e (b) agregação dos conjuntos inferidos.

O valor resultante é finalmente enviado para o jogo, que o utiliza para controlar o intervalo e os efeitos da animação do tesouro. Um exemplo do resultado final na animação do jogo é ilustrado pela Figura 16, na qual o valor inferido foi utilizado para redimensionar a área de incerteza onde encontra-se um tesouro e a distância entre o personagem do jogador, representado pela seta vermelha, e a área do tesouro. Como visto nos quadros 16a, 16b e 16c, a utilização do motor de inferência *fuzzy* garante que essa transição seja realizada gradualmente.

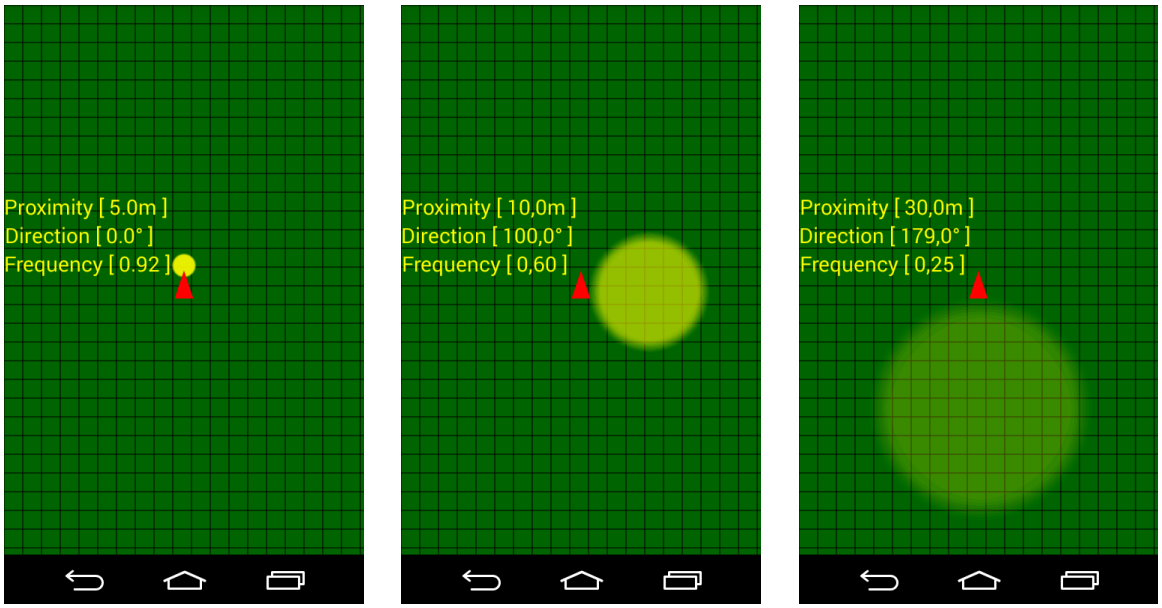


Figura 16 – Resultado da frequência na animação do jogo.

3.4 Considerações Finais

Nesse capítulo foram apresentados o *design* e a implementação de um jogo pervasivo móvel como prova de conceito da biblioteca proposta. O processo de implementação descreveu como as entidades da biblioteca, baseadas no modelo conceitual, podem ser

estendidas para modelar comportamentos específicos de um jogo pervasivo em troca da generalidade, e o processo de inferência apresentou como os dados contextuais podem ser tratados por meio de motores de inferência *fuzzy*.

4 Estudo Experimental

Este capítulo apresenta o estudo experimental cujo propósito é verificar se a biblioteca Gamepad proporciona algum benefício ao desenvolvedor de jogos pervasivos móveis no processo de manipulação dos dados contextuais. Dessa forma, na Seção 4.1 são detalhados os objetivos, planejamento e a condução da avaliação experimental realizada. Em seguida, os resultados do experimento são apresentados e discutidos na Seção 4.2. Na Seção 4.3 são levantadas as ameaças à validade desse estudo, e retratadas as medidas tomadas para mitigá-las. Por fim, a Seção 4.4 apresenta os estudos conduzidos previamente, que contribuíram na formulação do estudo atual.

4.1 Definição do Estudo

O principal objetivo deste estudo experimental é avaliar a biblioteca Gamepad como uma ferramenta adequada ao desenvolvimento dos JPMs. Para isso, buscou-se analisar a implementação de funcionalidades dos JPMs, com e sem a utilização da biblioteca Gamepad, com a intenção de verificar o efeito dessa ferramenta nas soluções desenvolvidas; com respeito ao tempo total de implementação, a extensão do código-fonte, sua complexidade e a facilidade de implementar a solução no ponto de vista do desenvolvedor. O estudo foi realizado no contexto acadêmico, com estudantes de graduação e pós-graduação. As subseções seguintes detalham o plano experimental.

4.1.1 Questões e métricas

As questões de pesquisa que norteiam este estudo são apresentadas a seguir:

QP1: *O tamanho do código-fonte que utiliza a biblioteca proposta é menor do que o tamanho do código-fonte produzido sem a utilização da biblioteca?* A extensividade do código-fonte serve como indicador do esforço requerido pela API. Neste caso, a quantidade de **linhas lógicas** do código-fonte foi utilizada como métrica;

QP2: *A complexidade do código-fonte da solução que utiliza a biblioteca proposta é menor do que a complexidade do código-fonte produzido sem a utilização da biblioteca?* Com base na complexidade do código-fonte, é possível presumir a manutenabilidade da API. A medida utilizada foi a **complexidade condicional** (MCCABE, 1976);

QP3: *O tempo necessário para implementar as tarefas do experimento sem a utilização da biblioteca proposta é superior ao tempo exigido pela implementação das mesmas, com a utilização da biblioteca?* Essa informação é um indicador do esforço e

produtividade das soluções produzidas com a API. A métrica utilizada foi a **duração** em segundos das tarefas de implementação;

QP4: *A implementação das tarefas com a biblioteca proposta é percebida pelo desenvolvedor como mais simples do que a implementação das tarefas sem a biblioteca?* O intuito dessa questão é acrescentar um elemento subjetivo que permita indagar os participantes sobre a usabilidade da API. Devido à subjetividade dessa questão, a medição foi realizada por meio de questionários.

4.1.2 Hipóteses

Em resposta às questões apresentadas, foram formuladas as seguintes hipóteses:

- **Tamanho**
  Hipótese nula ($H_0^{tamanho}$): $\mu_{l_{CB}} = \mu_{l_{SB}}$, isto é, o *tamanho* exibido pelas implementações *com a biblioteca* proposta ($\mu_{l_{CB}}$) *não é* significantemente *diferente do tamanho* exibido pelas implementações *sem a biblioteca* ($\mu_{l_{SB}}$);
  Hipótese alternativa ($H_1^{tamanho}$): $\mu_{l_{CB}} \neq \mu_{l_{SB}}$;

- **Complexidade**
  Hipótese nula ($H_0^{complexidade}$): $\mu_{c_{CB}} = \mu_{c_{SB}}$, isto é, a *complexidade* exibida pelas implementações *com a biblioteca* proposta ($\mu_{c_{CB}}$) *não é* significantemente *diferente da complexidade* exibida pelas implementações *sem a biblioteca* ($\mu_{c_{SB}}$);
  Hipótese alternativa ($H_1^{complexidade}$): $\mu_{c_{CB}} \neq \mu_{c_{SB}}$;

- **Tempo**
  Hipótese nula (H_0^{tempo}): $\mu_{t_{CB}} = \mu_{t_{SB}}$, isto é, o *tempo* exibido pelas implementações *com a biblioteca* proposta ($\mu_{t_{CB}}$) *não é* significantemente *diferente do tempo* exibido pelas implementações *sem a biblioteca* ($\mu_{t_{SB}}$);
  Hipótese alternativa (H_1^{tempo}): $\mu_{t_{CB}} \neq \mu_{t_{SB}}$;

- **Qualidade**
  Hipótese nula ($H_0^{qualidade}$): $\mu_{q_{CB}} = \mu_{q_{SB}}$, isto é, a *qualidade* percebida nas soluções implementadas *com a biblioteca* proposta *não é* significantemente *diferente da qualidade* percebida nas soluções implementadas *sem a biblioteca*;
  Hipótese alternativa ($H_1^{qualidade}$): $\mu_{q_{CB}} \neq \mu_{q_{SB}}$.

4.1.3 Definição das variáveis

As **variáveis independentes** são os fatores que podem ser controlados antes do experimento, e representam a causa dos efeitos observados pelos resultados. Neste estudo, essas variáveis foram: (i) a linguagem de programação e a plataforma selecionada, Kotlin

e Android; (*ii*) e as ferramentas de desenvolvimento selecionadas para implementar as tarefas do experimento, o SDK padrão da plataforma Android e a biblioteca Gamepad.

Em consequência, as **variáveis dependentes** são aquelas que refletem o efeito das variáveis independentes. Neste estudo, essas variáveis foram: (*i*) o tempo gasto para concluir a implementação da tarefa (*time*); (*ii*) a quantidade de linhas lógicas no código-fonte da solução (*lloc*); (*iii*) a complexidade condicional do código-fonte implementado (*mcc*); (*iv*) e a percepção de simplicidade da implementação e compreensão da solução (*ql*).

4.1.4 Contexto e instrumentação

O experimento foi realizado em um ambiente laboratorial de maneira controlada, o que caracteriza uma execução *offline* (*in vitro*). Entre os dez (10) participantes do experimento, a maior parte (9) eram estudantes universitários de graduação, com apenas um (1) estudante de pós-graduação. Visto que foi realizada uma comparação entre duas abordagens, uma pré-existente e outra abordagem alternativa, esse experimento pode ser considerado de caráter específico (WOHLIN et al., 2012). Como esse tipo de estudo comparativo pode afetar o comportamento do participante em resposta aos tratamentos, isto será melhor discutido na Seção 4.3.

Como objeto do estudo para a aplicação dos tratamentos, decidiu-se realizar atividades práticas de implementação. A prática consiste em implementar os procedimentos para coletar, processar e utilizar dados contextuais dos sensores indicados para interagir com os controles de um jogo para dispositivos móveis. Dessa forma, o principal instrumento deste experimento foi o projeto para a plataforma Android contemplando o estado inicial da solução e um conjunto de instruções que descrevem como alcançar o resultado final.[1]

Como instrumento de medição, utilizou-se a ferramenta de análise estática *Detekt*[2] para obter as métricas a partir do código-fonte na linguagem Kotlin. Questionários pré-experimento e pós-experimento, que foram anexados no Apêndice A, foram utilizados para coletar dados de caracterização dos participantes e a opinião dos participantes sobre o experimento e as tarefas implementadas. Além disso, as instruções e manuais também foram disponibilizadas aos participantes na forma impressa e digital.

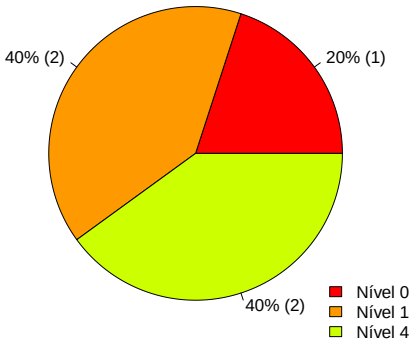
4.1.5 Seleção dos participantes

O recrutamento de participantes para este experimento provou-se uma tarefa dificultosa entre os meses de Dezembro a Janeiro, fora do período letivo. Por este motivo, os participantes foram selecionados de acordo com a conveniência, o que caracteriza este experimento como um *quasi-experiment* (WOHLIN et al., 2012).

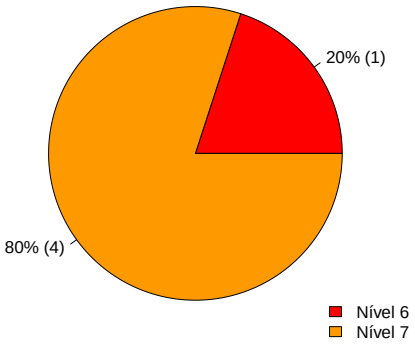
[1] O projeto completo pode ser encontrado no endereço: <https://github.com/vekat/SpaceEvader>
[2] Projeto de código aberto da ferramenta *Detekt*: <https://github.com/arturbosch/detekt>

A maior parte dos participantes selecionados para o experimento foi composta por alunos do curso de Ciência da Computação (DC/UFPI), que já haviam cursado as disciplinas de Programação Orientada a Objetos. Visto que a linguagem Kotlin, utilizada pela biblioteca Gamepad, é uma adição recente da plataforma Android, decidiu-se ofertar aos alunos um minicurso com carga horária de 8 horas sobre o desenvolvimento de aplicativos Android utilizando a linguagem Kotlin para contextualizar os participantes.

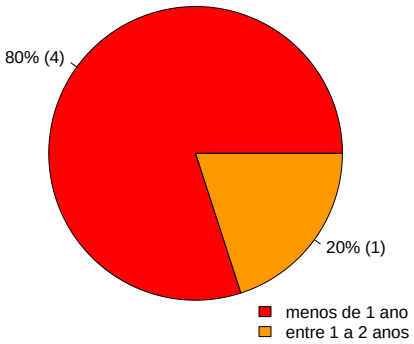
Antes da execução do experimento, os participantes foram entrevistados por meio de um questionário *online* para obter características como o nível de conhecimento nas ferramentas que viriam a ser utilizadas no experimento. Este questionário de caracterização encontra-se no Apêndice A.1. Entre os 10 participantes do experimento, apenas 5 concluíram todas as etapas. Os resultados do questionário de caracterização são exibidos na Figura 17.



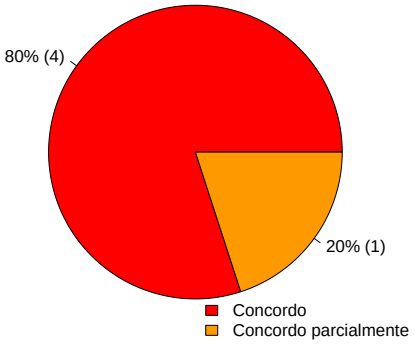
(a) Conhecimento em Kotlin



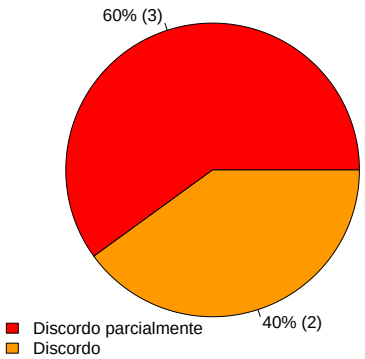
(b) Conhecimento em Java



(c) Experiência com Android



(d) Facilidade com bibliotecas



(e) Dificuldade com linguages

Figura 17 – Caracterização dos participantes do experimento.

4.1.6 Desenho experimental

O desenho experimental determina como os tratamentos são aplicados aos fatores em estudo. Para este estudo experimental, foi adotado um único fator, a implementação de funcionalidades dos JPM, e dois tratamentos, com e sem o uso da biblioteca Gamepad, aplicados a um grupo único, conforme apresentado pela Tabela 4.

Tabela 4 – Desenho experimental utilizado na avaliação da biblioteca Gamepad.

	Implementação das funcionalidades de um JPM			
Grupos	**1ª Etapa**		**2ª Etapa**	
Grupo Único	Treinamento	Implementação com a biblioteca	Treinamento	Implementação sem a biblioteca

Esse desenho foi baseado no desenho experimental utilizado por Oliveira (2016), que busca reduzir o risco de ameaças relacionadas a questões sociais, mas em contra partida, aumenta o risco da maturação afetar o experimento. A execução do estudo aconteceu em duas etapas. Na primeira, os participantes implementam os controles pervasivos com o uso da biblioteca Gamepad. Na segunda, os mesmos procedimentos são realizados em um novo projeto, mas sem o apoio da biblioteca. Os testes estatísticos aplicáveis a este tipo de desenho experimental são o teste paramétrico *T-test*, e o teste não-paramétrico *Wilcoxon-Mann-Whitney* (WOHLIN et al., 2012). Ambos os testes são tolerantes a um baixo número de amostras ($n \geq 5$) quando comparam distribuições pareadas.

4.1.7 Operação

Inicialmente, foi realizada uma apresentação aos participantes sobre as tarefas de implementação. No início da primeira e segunda etapa do experimento, foi entregue e apresentado o documento guia (ver Apêndice A.3) contendo a lista de passos a serem realizados para a conclusão da tarefa. Em seguida, os participantes foram informados sobre o procedimento a ser realizado quando a atividade fosse implementada. Ao final da segunda etapa, os participantes foram convidados a responder ao questionário de pós-experimento, apresentado no Apêndice A.2. Parte das perguntas apresentadas nesse questionário buscaram medir o que os participantes sentiram em relação à simplicidade de implementação e a facilidade de entender as APIs das bibliotecas abordadas (PICCIONI; FURIA; MEYER, 2013).

Dos 10 participantes, cinco foram eliminados por não terem concluído todas as etapas do experimento. Dois participantes abandonaram o experimento durante a 1ª etapa. Os outros três participantes concluíram o minicurso, mas não puderam comparecer ao experimento e não foi possível reagendar uma nova execução. Os dados dos cinco

participantes restantes foram utilizados para obter os resultados deste estudo e são apresentados na próxima seção.

4.2 Resultados e Discussões

Os resultados obtidos do experimento para as variáveis dependentes *lloc*, *mcc* e *time* são apresentados pelas Tabelas 5, 6 e 7, respectivamente. As análises realizadas nesta seção foram executadas usando a versão 3.2.3 do *software* R (R CORE TEAM, 2013). Verificando as distribuições, é possível perceber que os resultados alcançados no primeiro tratamento (com a biblioteca) são sempre menores, em média, do que os resultados obtidos no segundo tratamento.

Tabela 5 – Resultados do número de linhas lógicas do código-fonte.

Linhas lógicas do código-fonte (*lloc*)							
Tratamento	**Participante**					**Média**	**Desvio padrão**
Gamepad	237	246	240	237	237	239.40	3.91
Padrão	258	264	264	258	264	261.59	2.28

Ao observar as distribuições das duas primeiras variáveis, *lloc* e *mcc*, também é possível perceber que não existem intersecções entre os valores apanhados com a biblioteca Gamepad e os valores obtidos com a API padrão. Essa característica pode ser vista como um indicador de que essas distribuições são diferentes.

Tabela 6 – Resultados da complexidade do código-fonte.

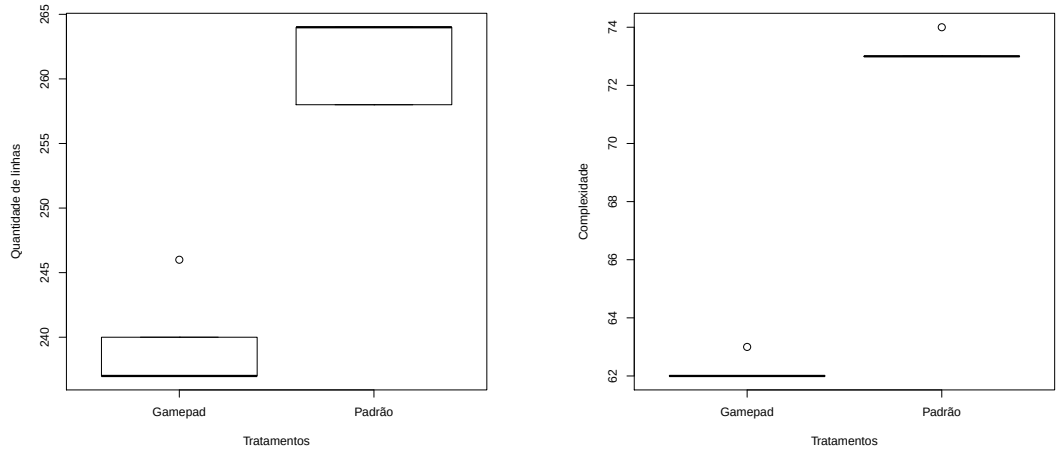
Complexidade de McCabe (*mcc*)							
Tratamento	**Participante**					**Média**	**Desvio padrão**
Gamepad	62	63	62	62	62	62.19	0.44
Padrão	73	73	74	73	73	73.20	0.44

Já para as distribuições da variável *time*, pode-se perceber uma intersecção. Os valores associados à biblioteca Gamepad apresentam uma maior variação em relação aos valores da abordagem padrão. Em ambas as distribuições, o participante 5 apresentou o menor tempo de implementação, sendo este um dos participantes que apontou ter mais experiência com a linguagem Kotlin. Os dados das três variáveis dependentes são ilustrados pelos diagramas de caixa (*box-plot*) nas Figuras 18a, 18b e 18c.

Antes de realizar os testes estatísticos com as distribuições, foi realizada uma análise da normalidade das amostras a fim de não violar as premissas desses testes. Na Figura 19, é ilustrada a densidade das distribuições de cada variável por tratamento. A partir desses gráficos, é possível perceber que o formato da curva para as variáveis *lloc* e *time* tem o

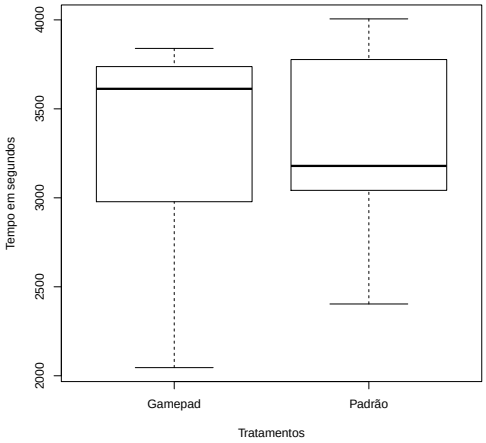
Tabela 7 – Resultados do tempo de experimento.

Tempo de implementação (*time*)							
Tratamento	**Participante**					**Média**	**Desvio padrão**
Gamepad	2978.1	3737.4	3612.5	3839.5	2045.7	3242.6	748.41
Padrão	3041.9	4005.5	3777.0	3179.2	2403.4	3281.4	634.14



(a) *lloc*

(b) *mcc*



(c) *time*

Figura 18 – *Box-plots* das distribuições das variáveis *lloc*, *mcc* e *time*.

formato de "sino", esperado de uma distribuição normal. Para verificar essa suposição, as distribuições foram submetidas ao teste de normalidade Shapiro-Wilk (SHAPIRO; WILK, 1965). Este teste é preferível por possuir poucas restrições em relação ao número mínimo de amostras. A implementação do teste Shapiro-Wilk encontra-se no Apêndice B.1.

Os resultados da métrica W do teste de Shapiro-Wilk são exibidos na Tabela 8. Este valor foi comparado com o valor crítico apresentado por Shapiro e Wilk (1965) ($W \geq 0.762$, quando $\alpha = 0.05$ e $n = 5$) para determinar se a distribuição é normal ou não. Dessa forma, é possível afirmar com um grau de confiança de 95% que apenas as distribuições da variável Tempo (*time*) são aproximadamente normais. Logo, para avaliar a terceira

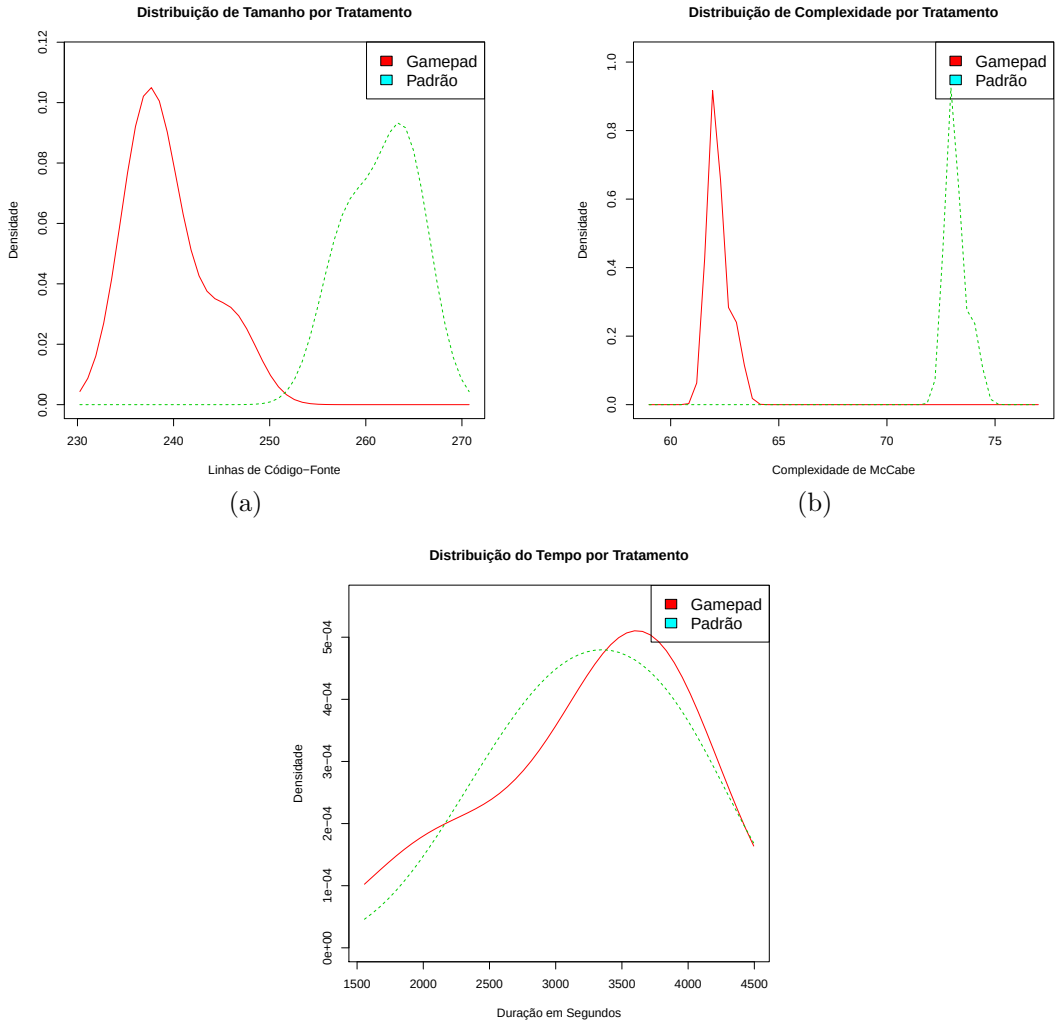


Figura 19 – Densidade das distribuições das variáveis *lloc*, *mcc* e *time*.

hipótese, relacionada à variável *time*, foi escolhido utilizar o método paramétrico teste-t para amostras pareadas (WINTER, 2013). Como as distribuições das variáveis *lloc* e *mcc* não aproximam-se a uma distribuição normal, foi decidido utilizar o teste não-paramétrico para amostras pareadas de Wilcoxon (ou *Wilcoxon-Mann-Whitney signed rank test*) para avaliar as duas primeiras hipóteses deste estudo (WILCOXON, 1945).

Tabela 8 – Resultados do teste de normalidade.

Variável	**Tratamento**	**Valor W**	***p-value***	**Normal**
lloc	Gamepad	0.73	0.0213	Não
	Padrão	0.68	0.0064	Não
mcc	Gamepad	0.55	0.0001	Não
	Padrão	0.55	0.0001	Não
time	Gamepad	**0.84**	0.1777	**Sim**
	Padrão	**0.95**	0.8031	**Sim**

A partir dos resultados do teste de Wilcoxon, é possível determinar se duas distribuições pareadas apresentam uma diferença estatisticamente significante por meio da comparação do valor V, métrica gerada pelo teste, com o valor crítico adequado (valor crítico: $V \leq 0$, quando $n = 5$ e $\alpha = 0.10$) (SANI; TODMAN, 2008). No entanto, não são determinados valores críticos para amostras com apenas cinco pares nos níveis de significância inferiores a $\alpha = 0.05$, tornando necessária a redução do grau de confiança deste estudo para 90%. Os resultados do teste Wilcoxon são apresentados pela Tabela 9. A implementação deste teste pode ser encontrada no Apêndice B.2.

Tabela 9 – Resultados do teste estatístico não-paramétrico.

Variável	**Valor V**	***p-value***	***Effect size***	**Diferentes**
lloc	**0**	0.0579	-0.8480	**Sim**
mcc	**0**	0.0544	-0.8600	**Sim**

Para os dados da variável Tamanho (*lloc*), o teste de Wilcoxon retornou um valor V igual a 0, e um *p-value* igual a 0.0579. Dessa forma, é possível afirmar com 90% de certeza ($\alpha = 0.10$) que as duas distribuições são estatisticamente diferentes. Portanto, a hipótese $H_0^{tamanho}$ foi rejeitada e a resposta à Questão de Pesquisa 1 é: *no contexto deste estudo, a quantidade de linhas de código necessárias para implementar funcionalidades de JPMs com a utilização da biblioteca Gamepad é, de fato, menor do que a quantidade exigida para implementar as mesmas funcionalidades sem o apoio da biblioteca.*

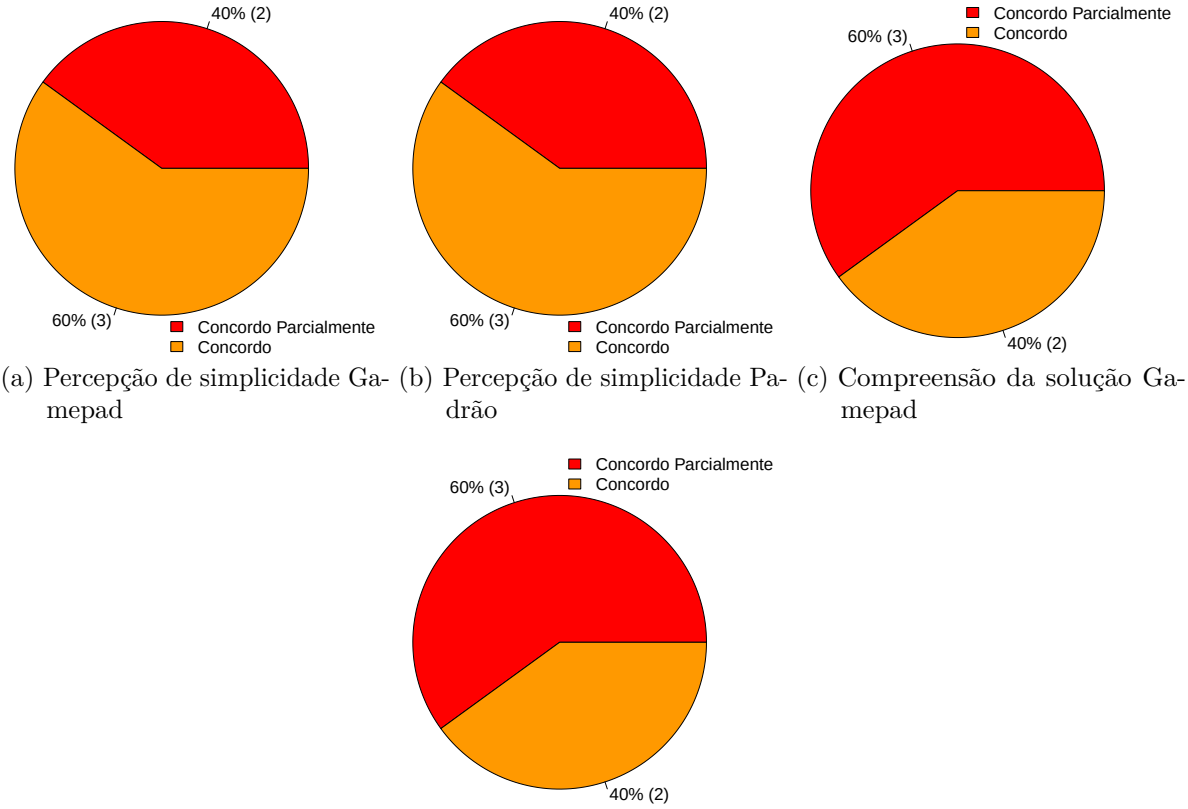
De maneira semelhante, o teste retornou um valor V igual a 0 e um *p-value* igual a 0.0544 para a variável Complexidade (*mcc*). Dessa forma, é possível afirmar com 90% de certeza ($\alpha = 0.10$) que as duas distribuições são estatisticamente diferentes. Portanto, a hipótese $H_0^{complexidade}$ foi rejeitada e a resposta à Questão de Pesquisa 2 é: *no contexto deste estudo, a complexidade do código-fonte produzido para implementar funcionalidades de JPMs com a utilização da biblioteca Gamepad é, de fato, menor do que a complexidade para implementar as mesmas funcionalidades sem o apoio da biblioteca.*

Já para a variável Tempo (*time*), o teste-t apontou, com nível de significância de 95% ($\alpha = 0.05$), que as duas distribuições não são estatisticamente diferentes, apresentando um *p-value* igual a 0.84. Os resultados do teste-t são apresentados pela Tabela 9. Assim, a hipótese H_0^{tempo} não pôde ser rejeitada e a resposta à Questão de Pesquisa 3 é: *o tempo necessário para implementar funcionalidades de JPMs com a utilização da biblioteca Gamepad não é menor do que o tempo para implementar as mesmas funcionalidades sem o apoio da biblioteca.*

Tabela 10 – Resultados do teste estatístico paramétrico.

Variável	**Estatística T**	***p-value***	**Diferentes**
time	-0.2135	0.8414	Não

Por fim, a Questão de Pesquisa 4 foi respondida com a análise dos dados coletados pelo questionário de pós-experimento (ver Apêndice A.2). Todos os participantes (100%) responderam que o experimento foi bem executado. No entanto, quanto à opinião dos participantes sobre a facilidade de uso e o entendimento da solução implementada, não houveram diferenças entre as notas da biblioteca Gamepad e o tratamento base. Dessa forma, a hipótese $H_0^{qualidade}$ não pôde ser rejeitada e a resposta à Questão de Pesquisa 4 é: *no contexto deste estudo e com base na opinião dos participantes, a qualidade das soluções criadas com a utilização da biblioteca Gamepad não é maior do que a qualidade das soluções feitas sem o apoio da biblioteca.* A Figura 20 resume os resultados obtidos no questionário.



(a) Percepção de simplicidade Gamepad

(b) Percepção de simplicidade Padrão

(c) Compreensão da solução Gamepad

(d) Compreensão da solução Padrão

Figura 20 – Resultados do questionário de pós-experimento.

4.3 Ameaças à Validade

Os resultados de qualquer experimento estão sujeitos a serem afetados por fatores indesejados, a ponto de prejudicarem sua validade. Dependendo dos objetivos do experimento, diferentes tipos de ameaças devem ser antecipados e priorizados para prevenir que a validade dos resultados seja afetada drasticamente (WOHLIN et al., 2012). Neste estudo experimental, foram priorizadas quatro tipos de ameaças à validade, na seguinte ordem: validade interna; validade de construção; validade da conclusão; e validade externa.

A validade interna do experimento é assegurada pela forte relação de causa e efeito entre os tratamentos e resultados. Dessa forma, é necessário prevenir que fatores externos desconhecidos afetem esse relacionamento. Para este experimento, a seleção dos participantes — alunos do minicurso sobre a linguagem Kotlin — pode ser considerada uma ameaça, pois não foi realizada uma aleatorização e os alunos podem sentir-se pressionados pela relação com o minicurso. Por isso, os participantes foram informados que a conclusão do minicurso não seria condicionada ao seu desempenho no experimento e que o mesmo não tratava-se de uma competição. Ameaças como a maturação e mortalidade foram amenizadas com a utilização de tarefas simplificadas. A maturação, particularmente, foi empregada contra a ferramenta Gamepad de maneira proposital para reduzir o custo do experimento, baseado na abordagem utilizada por Oliveira (2016). No entanto, é possível que essa suposição não corresponda ao comportamento observado entre os participantes deste experimento como, por exemplo, um participante pode sentir-se desmotivado com a realização de tarefas similares, afetando negativamente o seu desempenho.

A validade de construção preocupa-se com o grau de relação entre a teoria do estudo e os resultados observados. Neste experimento, uma ameaça à validade de construção é se o desenho experimental escolhido, realizado com grupo único para reduzir o custo, pode ser considerado confiável para medir o efeito dos tratamentos. Outras ameaças à validade de construção não foram consideradas significantes. O pesquisador não auxiliou os participantes durante o experimento com dúvidas sobre os tratamentos, e estes não foram informados sobre as hipóteses do experimento, ou que uma das ferramentas fora desenvolvida pelo pesquisador e, portanto, acredita-se que os participantes não foram influenciados a adivinhar os resultados esperados.

A validade da conclusão pode ser prejudicada por questões relacionadas à análise estatística dos resultados e a composição da amostra. Neste experimento, os testes estatísticos seguiram as recomendações para o desenho experimental aplicado, apresentadas em Wohlin et al. (2012). Para evitar a violação de premissas desses testes, também foi realizada uma análise visual da normalidade e estatística de cada distribuição da amostra. No entanto, o baixo número de participantes pode ser considerado uma ameaça à validade da conclusão, pois isto dificulta a observação de padrões na amostra e pode prejudicar a confiabilidade de certos testes estatísticos.

Por fim, a validade externa trata da capacidade de generalizar os resultados do experimento para outros ambientes. Considerando que o público alvo da biblioteca proposta são pesquisadores e desenvolvedores de jogos pervasivos móveis, a maior ameaça à validade externa é o fato de que os participantes do experimento são estudantes de graduação e pós-graduação sem relação com esta área de pesquisa. Inicialmente, foi considerado recrutar apenas participantes com familiaridade sobre o tema, porém, não foi houve quantidade suficiente de participantes para permitir uma filtragem. Além disso, é possível argumentar

que o SDK padrão da plataforma Android não é necessariamente a ferramenta utilizada no ambiente industrial, o que limita a generalização dos resultados para esse meio.

4.4 Estudos Anteriores

Antes da execução deste estudo experimental, foi realizada uma discussão a respeito do design da API da biblioteca entre o autor deste trabalho e um desenvolvedor de jogos móveis. Nessa discussão, foram identificadas e corrigidas algumas falhas da API que poderiam confundir o desenvolvedor. Essa experiência reforçou a motivação de realizar um grupo focal com desenvolvedores de jogos pervasivos. Em seguida, foi realizado um estudo piloto com dois participantes para obter *feedback* sobre o material do experimento. Após a execução do estudo piloto, os documentos de treinamento foram reformulados para apresentarem mais exemplos de código, pois os participantes expressaram que alguns passos estavam confusos. Também foi realizado um experimento anterior ao estudo experimental apresentado por este capítulo. No entanto, a primeira etapa estendeu-se por mais tempo do que havia sido previsto, o que impossibilitou a conclusão da segunda etapa do experimento para uma parte significante dos participantes. Esse evento ocasionou um atraso nas atividades planejadas, pois o plano experimental, em especial os objetos de estudo, precisou ser reformulado para que os mesmos participantes pudessem ser convidados para uma nova execução do experimento. Este atraso resultou no cancelamento do grupo focal previsto para ser realizado com alguns participantes do experimento.

4.5 Considerações Finais

Esse capítulo apresentou um estudo empírico para validar as funcionalidades da biblioteca Gamepad por meio de uma comparação das implementações de jogos pervasivos móveis. Foram considerados quatro critérios nessa comparação: a duração do desenvolvimento da solução; a quantidade de linhas do código-fonte; a complexidade do código-fonte; e a percepção da qualidade das soluções coletada por meio de questionários.

Os resultados do experimento sugerem que a biblioteca Gamepad pode ajudar a reduzir o tamanho e a complexidade dos JPM graças à sua API, porém, o estudo apontou que não existem diferenças significantes para o tempo de desenvolvimento e a percepção de qualidade entre as soluções produzidas com e sem a Gamepad. Além disso, devido ao alto número de desistências, a significância estatística desses resultados foi reduzida para 90% ($\alpha = 0.10$), diminuindo a confiança das conclusões. Diante dessas limitações, a realização de novos estudos empíricos, inclusive um Estudo de Caso, é recomendável (WOHLIN et al., 2012). Essa proposta será discutida na seção de Trabalhos Futuros.

5 Conclusões e Trabalhos Futuros

Visando apoiar o desenvolvedor de jogos pervasivos móveis a lidar com as incertezas inerentes aos dados contextuais, este trabalho apresentou uma biblioteca para abstrair o processo de manipulação destes dados. A abordagem empregada pela biblioteca buscou aproximar a forma como os dados contextuais são manipulados pelo desenvolvedor, com o procedimento utilizado pelo sistema de tratamento das entradas nos jogos digitais convencionais. Esta aproximação foi alcançada por meio de uma representação alternativa para os dados contextuais, que foi especificada por um modelo conceitual, utilizado como base para a implementação da API da biblioteca, nomeada Gamepad.

Para demonstrar a aplicabilidade da biblioteca, um jogo pervasivo móvel de caça-ao-tesouro, chamado Radar, foi desenvolvido e apresentado como prova de conceito. Sua implementação empregou dois objetos do tipo `Fonte`, um objeto `Botão` e um `Interpretador` para realizarem a tarefa de inferir o valor de uma variável do jogo, denominada "intensidade", com base numa combinação dos dados contextuais concedidos pelos sensores GPS, acelerômetro e magnetômetro. O cerne desta tarefa foi executada pelo objeto `Interpretador`, que permitiu ao jogo aplicar um motor de inferência *fuzzy* para transformar os dados contextuais em conjuntos *fuzzy*, e inferir a "intensidade" com base num conjunto de regras *fuzzy*. Por fim, os valores inferidos, um para cada tesouro, foram usados para controlar a animação do jogo, sinalizando ao jogador o efeito de suas ações. Essa mesma lógica de jogo pode ser utilizada para desenvolver jogos de diferentes gêneros como, por exemplo, um jogo de batalha com cartas que utilize informações climáticas a respeito da região geográfica do jogador para alterar o poder das suas cartas durante uma partida.

Embora uma prova de conceito permita que as funcionalidades da biblioteca sejam postas a teste, este tipo de demonstração não é o suficiente para validar a biblioteca proposta. Assim, um estudo experimental foi planejado e conduzido, de acordo com recomendações da Engenharia de *Software* Experimental, para verificar se a biblioteca Gamepad oferece algum benefício ao desenvolvedor de jogos pervasivos móveis em relação ao tamanho, complexidade e tempo de desenvolvimento do código-fonte, e a qualidade da API. Os resultados do experimento indicaram que as soluções implementadas com o apoio da biblioteca Gamepad são menos extensas e mais simples, porém, não foram encontradas diferenças significativas no tempo de desenvolvimento e percepção de qualidade da API.

Por fim, a biblioteca Gamepad encontra-se disponível como um projeto de código-fonte aberto no endereço <https://github.com/vekat/gamepad>. Dessa forma, é possível que a biblioteca continue evoluindo com a ajuda de outros desenvolvedores e pesquisadores. A seguir, são apresentadas as limitações desta pesquisa e ideias para trabalhos futuros.

5.1 Limitações e Trabalhos Futuros

A seguir, são apontadas limitações desta pesquisa que poderão ser abordadas em trabalhos futuros:

- **Novas Funcionalidades**: atualmente a biblioteca Gamepad implementa objetos `Fonte` para subconjunto dos sensores mais comuns da plataforma Android, e apenas uma implementação do objeto `Interpretador` para FISs do Tipo 1. Dessa forma, seria positivo separar essas implementações do núcleo da biblioteca por meio de módulos Android, possibilitando ao desenvolvedor escolher somente aquelas funcionalidades julgadas necessárias em cada projeto. Essa separação facilitaria a utilização de objetos `Fonte`, `Botão` e `Interpretador` desenvolvidos por terceiros, que poderiam ser disponibilizados em repositórios da plataforma Android, como o JCenter;[1]

- **Realização de Grupos Focais**: como mencionado na Seção 4.4, a realização de um grupo focal estava prevista, no entanto, não foi possível conduzi-la devido à necessidade de reformular os objetos do estudo com a falha do primeiro experimento. A realização de grupos focais com os participantes do experimento seria uma maneira de coletar *feedback* direto a respeito da API da biblioteca, possibilitando a identificação de falhas no seu *design*, a fim de melhorar a usabilidade da biblioteca;

- **Mapeamento Sistemático**: com o propósito de fortalecer a justificativa da abordagem proposta, seria interessante realizar um mapeamento sistemático para identificar outras ferramentas que oferecem apoio ao desenvolvimento de jogos pervasivos móveis. O mapeamento sistemático seria uma forte contribuição a esta pesquisa, possibilitando a realização de comparações qualitativas e/ou quantitativas com outras ferramentas semelhantes;

- **Execução de Novos Estudos Empíricos**: como visto na Seção 4.2, a taxa de mortalidade do experimento foi alta (50%), o que causou uma diminuição no poder estatístico do estudo devido ao baixo número de amostras. Apesar dessa limitação, foram percebidos indícios de que a biblioteca realmente beneficia o desenvolvedor de jogos pervasivos móveis. No entanto, as conclusões deste estudo podem ser fortalecidas com a condução de um novo experimento, a fim de assegurar o poder estatístico dos resultados. Considerando os estudos anteriores como aprendizagem, é possível incrementar o plano experimental atual para aumentar o foco na usabilidade com base no estudo de Piccioni, Furia e Meyer (2013). Além disso, a execução de um Estudo de Caso com desenvolvedores de jogos no meio empresarial seria necessária para avaliar o efeito real dos benefícios da biblioteca (WOHLIN et al., 2012). Uma possibilidade de plano experimental revisado é apresentado no Apêndice C.

[1] Repositório JCenter: <https://bintray.com/bintray/jcenter>

Referências

ALEGRE, U.; AUGUSTO, J. C.; CLARK, T. Engineering context-aware systems and applications. *Journal of Systems and Software*, Elsevier Science Inc., New York, NY, USA, v. 117, n. C, p. 55–83, jul. 2016. ISSN 0164-1212. Citado 3 vezes nas páginas 1, 5 e 8.

ANAGNOSTOPOULOS, C.; HADJIEFTHYMIADES, S. Enhancing situation-aware systems through imprecise reasoning. *IEEE Transactions on Mobile Computing*, v. 7, n. 10, p. 1153–1168, out. 2008. ISSN 1536-1233. Citado na página 14.

BALDAUF, M.; DUSTDAR, S.; ROSENBERG, F. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, Inderscience Publishers, v. 2, n. 4, p. 263–277, 2007. Citado 3 vezes nas páginas 3, 14 e 16.

BAUMGARTNER, N.; GOTTESHEIM, W.; MITSCH, S.; RETSCHITZEGGER, W.; SCHWINGER, W. Beaware!—situation awareness, the ontology-driven way. *Data & Knowledge Engineering*, Elsevier, v. 69, n. 11, p. 1181–1193, 2010. Citado na página 14.

BECAM, A.; NENONEN, V. A. A. Designing and creating environment aware games. In: *2008 5th IEEE Consumer Communications and Networking Conference*. [S.l.: s.n.], 2008. p. 1045–1049. ISSN 2331-9852. Citado 3 vezes nas páginas 1, 13 e 25.

BENFORD, S.; ANASTASI, R.; FLINTHAM, M.; DROZD, A.; CRABTREE, A.; GREENHALGH, C.; TANDAVANITJ, N.; ADAMS, M.; ROW-FARR, J. Coping with uncertainty in a location-based game. *IEEE Pervasive Computing*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 2, n. 3, p. 34–41, jul. 2003. ISSN 1536-1268. Citado na página 13.

BETTINI, C.; BRDICZKA, O.; HENRICKSEN, K.; INDULSKA, J.; NICKLAS, D.; RANGANATHAN, A.; RIBONI, D. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 6, n. 2, p. 161–180, abr. 2010. ISSN 1574-1192. Citado 2 vezes nas páginas 2 e 13.

BJÖRK, S.; HOLOPAINEN, J.; LJUNGSTRAND, P.; MANDRYK, R. Special issue on ubiquitous games. *Personal and Ubiquitous Computing*, v. 6, n. 5, p. 358–361, dez. 2002. ISSN 1617-4909. Disponível em: <https://doi.org/10.1007/s007790200040>. Citado 2 vezes nas páginas 1 e 9.

BLANCHETTE, J. The little manual of api design. 2008. Disponível em: <https://people.mpi-inf.mpg.de/~jblanche/api-design.pdf>. Citado 2 vezes nas páginas 10 e 15.

BLOCH, J. How to design a good api and why it matters. In: *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*. New York, NY, USA: ACM, 2006. (OOPSLA '06), p. 506–507. ISBN 1-59593-491-X. Disponível em: <http://doi.acm.org/10.1145/1176617.1176622>. Citado 2 vezes nas páginas 15 e 20.

VAN BROEKHOVEN, E.; DE BAETS, B. Fast and accurate center of gravity defuzzification of fuzzy system outputs defined on trapezoidal fuzzy partitions. *Fuzzy Sets and Systems*, Elsevier North-Holland, Inc., Amsterdam, The Netherlands, The Netherlands, v. 157, n. 7, p. 904–918, abr. 2006. ISSN 0165-0114. Disponível em: <http://dx.doi.org/10.1016/j.fss.2005.11.005>. Citado na página 31.

BUTLER, M. Android: Changing the mobile landscape. *IEEE Pervasive Computing*, v. 10, n. 1, p. 4–7, jan. 2011. ISSN 1536-1268. Citado na página 20.

CHALMERS, M.; BELL, M.; HALL, M.; SHERWOOD, S.; TENNENT, P. Seamful games. 2004. Citado 2 vezes nas páginas 2 e 13.

DEY, A. K. Understanding and using context. *Personal and Ubiquitous Computing*, Springer-Verlag, London, UK, v. 5, n. 1, p. 4–7, jan. 2001. ISSN 1617-4909. Citado 4 vezes nas páginas 1, 7, 8 e 17.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, v. 16, n. 2, p. 97–166, dez. 2001. ISSN 0737-0024. Citado 4 vezes nas páginas 12, 14, 15 e 16.

DEY, A. K.; SALBER, D.; ABOWD, G. D.; FUTAKAWA, M. *An Architecture to Support Context-Aware Applications*. [S.l.], 1999. Citado 3 vezes nas páginas 5, 7 e 12.

DUBOIS, D.; PRADE, H. M. *Fuzzy Sets and systems: Theory and Applications*. [S.l.]: Academic press, 1980. v. 144. (Mathematics in science and engineering, v. 144). Citado na página 10.

ECONOMOU, D.; BOUKI, V.; KOUNENIS, T.; MENTZELOPOULOS, M.; GEORGALAS, N. Treasure hunt pervasive games in cultural organisations. In: *2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)*. [S.l.: s.n.], 2015. p. 368–372. Citado na página 25.

ENGELBRECHT, A. P. *Computational Intelligence: An Introduction*. [S.l.]: John Wiley & Sons, 2007. Citado na página 11.

FERREIRA, D.; KOSTAKOS, V.; DEY, A. K. AWARE: Mobile context instrumentation framework. *Frontiers in ICT*, Frontiers, v. 2, p. 1–9, 2015. Citado 4 vezes nas páginas 12, 14, 15 e 16.

GARLAN, D.; SHAW, M. *An Introduction to Software Architecture*. Pittsburgh, PA, USA, 1994. Citado 2 vezes nas páginas 5 e 9.

HAGHIGHI, P. D.; KRISHNASWAMY, S.; ZASLAVSKY, A.; GABER, M. M. Reasoning about context in uncertain pervasive computing environments. In: *Proceedings of the 3rd European Conference on Smart Sensing and Context*. Berlin, Heidelberg: Springer-Verlag, 2008. (EuroSSC '08), p. 112–125. ISBN 978-3-540-88792-8. Citado na página 14.

HENNING, M. Api design matters. *Communications of the ACM*, ACM, New York, NY, USA, v. 52, n. 5, p. 46–56, maio 2009. ISSN 0001-0782. Citado 2 vezes nas páginas 10 e 15.

JETBRAINS. *Kotlin Language Documentation*. [S.l.], 2018. Disponível em: <https://kotlinlang.org/docs/kotlin-docs.pdf>. Citado na página 20.

KASAPAKIS, V.; GAVALAS, D. Pervasive gaming: Status, trends and design principles. *Journal of Network and Computer Applications*, Elsevier, v. 55, p. 213–236, 2015. Citado 3 vezes nas páginas 1, 2 e 12.

KASAPAKIS, V.; GAVALAS, D. Investigating the effect of user generated content in pervasive games. In: *2016 IEEE Symposium on Computers and Communication (ISCC)*. [S.l.: s.n.], 2016. p. 79–84. Citado na página 15.

KRUCHTEN, P. The 4+1 view model of architecture. *IEEE Software*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 12, n. 6, p. 42–50, nov. 1995. ISSN 0740-7459. Disponível em: <http://dx.doi.org/10.1109/52.469759>. Citado na página 15.

LEI, H.; SOW, D. M.; DAVIS II, J. S.; BANAVAR, G.; EBLING, M. R. The design and applications of a context service. *SIGMOBILE Mobile Computing and Communications Review*, ACM, New York, NY, USA, v. 6, n. 4, p. 45–55, out. 2002. ISSN 1559-1662. Disponível em: <http://doi.acm.org/10.1145/643550.643554>. Citado na página 14.

LINNER, D.; KIRSCH, F.; RADUSCH, I.; STEGLICH, S. Context-aware multimedia provisioning for pervasive games. In: *Proceedings of the Seventh IEEE International Symposium on Multimedia*. Washington, DC, USA: IEEE Computer Society, 2005. (ISM '05), p. 60–68. ISBN 0-7695-2489-3. Citado na página 9.

MAIA, M. E. F.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. M. C. Loccam - loosely coupled context acquisition middleware. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2013. (SAC '13), p. 534–541. ISBN 978-1-4503-1656-9. Disponível em: <http://doi.acm.org/10.1145/2480362.2480465>. Citado 2 vezes nas páginas 15 e 16.

MCCABE, T. J. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2, n. 4, p. 308–320, dez. 1976. ISSN 0098-5589. Citado na página 33.

MONTOLA, M. Exploring the edge of the magic circle: Defining pervasive games. In: *Proceedings of Digital Arts and Culture*. Copenhagen, Denmark: [s.n.], 2005. p. 1–4. Citado 2 vezes nas páginas 1 e 9.

MONTOLA, M.; STENROS, J.; WAERN, A. *Pervasive Games: Theory and Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN 0123748534, 9780123748539. Citado na página 9.

MUKHOPADHYAY, S. C. Wearable sensors for human activity monitoring: A review. *IEEE Sensors Journal*, v. 15, n. 3, p. 1321–1330, mar. 2015. ISSN 1530-437X. Citado na página 26.

MYERS, B. A.; STYLOS, J. Improving api usability. *Communications of the ACM*, ACM, New York, NY, USA, v. 59, n. 6, p. 62–69, maio 2016. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/2896587>. Citado 2 vezes nas páginas 10 e 15.

NIEUWDORP, E. The pervasive discourse: An analysis. *Computers in Entertainment*, ACM, New York, NY, USA, v. 5, n. 2, abr. 2007. ISSN 1544-3574. Disponível em: <http://doi.acm.org/10.1145/1279540.1279553>. Citado 2 vezes nas páginas 1 e 8.

OLIVEIRA, P. A. M. *Athena: uma Arquitetura e uma Ferramenta para Auxiliar o Desenvolvimento de Sistemas Baseados em Inteligência Computacional*. Tese (mestrado) — Universidade Federal do Piauí, mar. 2016. Citado 2 vezes nas páginas 37 e 43.

PAAVILAINEN, J.; KORHONEN, H.; ALHA, K.; STENROS, J.; KOSKINEN, E.; MAYRA, F. The pokémon go experience: A location-based augmented reality mobile game goes mainstream. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2017. (CHI '17), p. 2493–2498. ISBN 978-1-4503-4655-9. Citado na página 2.

PEDRYCZ, W.; GOMIDE, F. *Fuzzy Systems Engineering: Toward Human-Centric Computing*. [S.l.]: Wiley-IEEE Press, 2007. ISBN 0471788570, 9780471788577. Citado na página 11.

PERERA, C.; ZASLAVSKY, A.; CHRISTEN, P.; GEORGAKOPOULOS, D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys Tutorials*, v. 16, n. 1, p. 414–454, jan. 2014. ISSN 1553-877X. Citado na página 8.

PICCIONI, M.; FURIA, C. A.; MEYER, B. An empirical study of api usability. In: *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2013. p. 5–14. ISSN 1949-3770. Citado 3 vezes nas páginas 10, 37 e 46.

R CORE TEAM. *R: A Language and Environment for Statistical Computing*. Vienna, Austria, 2013. Disponível em: <http://www.R-project.org/>. Citado na página 38.

RANGANATHAN, A.; AL-MUHTADI, J.; CAMPBELL, R. H. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, v. 3, n. 2, p. 62–70, abr. 2004. ISSN 1536-1268. Citado na página 14.

ROBERT-BOUCHARD, R.; DUPIRE, J.; CUBAUD, P. Designing indoor tangible games based on fuzzy localisation. In: *IEEE Games Entertainment Media Conference (GEM), 2015*. [S.l.: s.n.], 2015. p. 1–4. Citado 2 vezes nas páginas 2 e 13.

ROBILLARD, M. P. What makes apis hard to learn? answers from developers. *IEEE Software*, v. 26, n. 6, p. 27–34, nov. 2009. ISSN 0740-7459. Citado na página 10.

SALBER, D.; DEY, A. K.; ABOWD, G. D. The context toolkit: Aiding the development of context-enabled applications. In: *Proceedings of the Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 1999. (CHI '99), p. 434–441. ISBN 0-201-48559-1. Citado na página 12.

SALEN, K.; ZIMMERMAN, E. *Rules of Play: Game Design Fundamentals*. [S.l.]: The MIT Press, 2003. ISBN 0262240459, 9780262240451. Citado na página 9.

SANCHEZ, L.; LANZA, J.; OLSEN, R.; BAUER, M.; GIROD-GENET, M. A generic context management framework for personal networking environments. In: *2006 3rd Annual International Conference on Mobile and Ubiquitous Systems - Workshops*. [S.l.: s.n.], 2006. p. 1–8. Citado na página 8.

SANI, F.; TODMAN, J. *Experimental design and statistics for psychology: a first course*. [S.l.]: John Wiley & Sons, 2008. Citado na página 41.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. In: *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 1994. (WMCSA '94), p. 85–90. ISBN 978-0-7695-3451-0. Citado 2 vezes nas páginas 1 e 7.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, Oxford University Press, Biometrika Trust, v. 52, n. 3/4, p. 591–611, 1965. ISSN 00063444. Disponível em: <http://www.jstor.org/stable/2333709>. Citado na página 39.

VALENTE, L.; FEIJÓ, B.; LEITE, J. C. Features and checklists to assist in pervasive mobile game development. *Proceedings of SBGames*, SBC, 2013. Citado 3 vezes nas páginas 2, 9 e 13.

VALENTE, L.; FEIJÓ, B.; LEITE, J. C. Mapping quality requirements for pervasive mobile games. *Requirements Engineering*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 22, n. 1, p. 137–165, mar. 2017. ISSN 0947-3602. Citado 3 vezes nas páginas 8, 9 e 13.

VIANA, J. R. M.; VIANA, N. P.; TRINTA, F. A. M.; CARVALHO, W. V. d. A systematic review on software engineering in pervasive games development. In: *Brazilian Symposium on Computer Games and Digital Entertainment*. [S.l.: s.n.], 2014. p. 51–60. ISSN 2159-6654. Citado na página 11.

WEISER, M. The computer for the 21st century. *Scientific American*, Nature Publishing Group, v. 265, n. 3, p. 94–104, set. 1991. Citado 2 vezes nas páginas 1 e 7.

WEISER, M. Some computer science issues in ubiquitous computing. *Communications of the ACM*, ACM, New York, NY, USA, v. 36, n. 7, p. 75–84, jul. 1993. ISSN 0001-0782. Citado na página 7.

WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin*, International Biometric Society, Wiley, v. 1, n. 6, p. 80–83, 1945. ISSN 00994987. Disponível em: <http://www.jstor.org/stable/3001968>. Citado na página 40.

DE WINTER, J. C. Using the student's t-test with extremely small sample sizes. *Practical Assessment, Research & Evaluation*, v. 18, n. 10, 2013. Citado na página 40.

WOHLIN, C.; RUNESON, P.; HST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLN, A. *Experimentation in Software Engineering*. [S.l.]: Springer Publishing Company, Incorporated, 2012. ISBN 3642290434, 9783642290435. Citado 6 vezes nas páginas 35, 37, 42, 43, 44 e 46.

YÜRÜR, Ö.; LIU, C. H.; SHENG, Z.; LEUNG, V. C. M.; MORENO, W.; LEUNG, K. K. Context-awareness for mobile sensing: A survey and future directions. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 68–93, dez. 2016. ISSN 1553-877X. Citado 4 vezes nas páginas 3, 12, 15 e 16.

ZADEH, L. A. Fuzzy sets. *Information and control*, Elsevier, v. 8, n. 3, p. 338–353, 1965. Citado na página 10.

ZADEH, L. A. Fuzzy logic and approximate reasoning. *Synthese*, Springer, v. 30, n. 3, p. 407–428, 1975. Citado na página 10.

ZIEGLER, S.; WOODWARD, R. C.; IU, H. H. C.; BORLE, L. J. Current sensing techniques: A review. *IEEE Sensors Journal*, v. 9, n. 4, p. 354–376, abr. 2009. ISSN 1530-437X. Citado 2 vezes nas páginas 8 e 26.

Apêndices

APÊNDICE A – Materiais do Experimento

A.1 Questionário pré-experimento

1 Por favor, informe o seu **nome** e um **sobrenome**. *

Participante

2 **Participante**, por favor, informe um e-mail para contato. *

3 Ok **Participante**, qual o seu grau de **escolaridade**? *

- ○ Graduando
- ○ Mestrando
- ○ Doutorando
- ○ Graduado
- ○ Mestrado
- ○ Doutorado

4 Você tem **experiência** desenvolvendo aplicativos Android? *

Quantos **anos** de experiência?

- ○ menos de 1 ano
- ○ entre 1 e 2 anos
- ○ entre 3 e 4 anos
- ○ mais que 5 anos

Qual o seu grau de **conhecimento** na linguagem Kotlin? *

0	1	2	3	4	5	6	7	8	9
○	○	○	○	○	○	○	○	○	○

Desconheço a linguagem — Conhecimento avançado

E o seu grau de **conhecimento** com a linguagem Java? *

0	1	2	3	4	5	6	7	8	9
○	○	○	○	○	○	○	○	○	○

Desconheço a linguagem — Conhecimento avançado

7 OK **Participante**, estamos quase no fim.
Agora queremos o seu ponto de vista sobre as seguintes afirmações:

a. "Eu tenho **facilidade** de **aprender** novas **bibliotecas** de *software*, ***frameworks*** ou **APIs**." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

b. "Eu costumo ter **dificuldades** ao **aprender** novas **linguagens** de programação." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

Enviar

Nunca submeta palavras-passe! - Reportar abuso

A.2 Questionário pós-experimento

1 Primeiro, por favor informe o seu **nome** e **sobrenome**. *

O mesmo que foi informado no primeiro questionário.

Participante

2 Participante, qual nota você atribui à **execução** do experimento? *

- Ruim - O experimento não foi claro e foi mal conduzido.
- Razoável - O experimento foi claro, mas teve problemas na execução.
- Bom - O experimento foi claro e não houve problemas de execução.

3 Em relação a abordagem usada nas tarefas implementação, informe o seu ponto de vista sobre as seguintes afirmações:

a. "A solução da **tarefa prática** com a biblioteca **Gamepad** foi **simples** de implementar." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

b. "A solução da **tarefa prática** com a biblioteca **padrão** foi **simples** de implementar." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

c. "A **implementação** da **solução** na tarefa prática com a biblioteca **Gamepad** foi **fácil** de compreender." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

d. "A **implementação** da **solução** na tarefa prática com a biblioteca **padrão** foi **fácil** de compreender." *

- Discordo
- Discordo parcialmente
- Indiferente
- Concordo parcialmente
- Concordo

4 Questão aberta.

Participante, se você tiver algum comentário sobre o experimento, as ferramentas utilizadas, sobre a sua implementação ou qualquer *feedback*, fique a vontade para digitar no espaço abaixo.

Enviar

Nunca submeta palavras-passe! - Reportar abuso

A.3 Guia da Tarefa

Atividade prática de programação

O objetivo desta prática é adicionar ao jogo **SpaceEvader** uma forma de controlar os mecanismos do jogo por meio da orientação do dispositivo móvel. Para implementar essa funcionalidade, duas APIs serão utilizadas: A API da biblioteca **Gamepad**; e a API **padrão** do SDK da plataforma Android. Durante a prática, os participantes podem utilizar recursos online, como a documentação das respectivas APIs, bem como os documentos de treinamento de cada API: treinamento-android e treinamento-gamepad.

A sequência de passos a ser executada no projeto **SpaceEvader** é apresenta a seguir nesse documento. Além disso, o código fonte inicial do projeto apresenta comentários (`// TODO: ...`) indicando qual passo precisa ser implementado, e o projeto contém um conjunto de *testes de unidade* que podem ser executados em qualquer momento da atividade prática para verificar o progresso da implementação.

Tela do menu principal

Dentro do arquivo `MenuActivity.kt` :

1. **Declare as variáveis** necessárias para receber os dados sobre a **orientação** do dispositivo móvel (seção 1);

Dentro do arquivo `MenuView.kt` :

2. **Adicione** na classe `MenuView` **a interface** que habilita o recebimento **de eventos** dos sensores por meio de um *método callback* (seção 2);

De volta ao arquivo `MenuActivity.kt` :

3. **Inicialize e configure** as variáveis declaradas no *passo #1* dentro do método `onCreate()` da classe `MenuActivity` (seção 3);

4. Adicione o método `onResume()` e implemente os procedimentos necessários para **iniciar o recebimento de eventos** (seção 4);

5. Adicione o método `onPause()` e implemente os procedimentos necessários para **cancelar o recebimento de eventos** (seção 4);

Dentro do arquivo `MenuView.kt` :

6. **Implemente** dentro do **método callback** da interface adicionada no *passo #2* os procedimentos necessários **para obter** o vetor com **a orientação** do dispositivo, e utilize o **primeiro valor do vetor** (*azimuth*) como argumento para chamar o método `atualizarEixo()` (seção 5).

Tela do jogo

Dentro do arquivo `GameActivity.kt`:

7. **Declare as variáveis** necessárias para receber os dados sobre a **orientação** do dispositivo móvel (seção 1);

Dentro do arquivo `GameView.kt`:

8. **Adicione** na classe `GameView` **a interface** que habilita o recebimento **de eventos** dos sensores por meio de um *método callback* (seção 2);

De volta ao arquivo `GameActivity.kt`:

9. **Inicialize e configure** as variáveis declaradas no *passo #7* dentro do método `onCreate()` da classe `GameActivity` (seção 3);

10. Implemente no método `onResume()` os procedimentos necessários para **iniciar o recebimento de eventos** (seção 4);

11. Implemente no método `onPause()` os procedimentos necessários para **cancelar o recebimento de eventos** (seção 4);

Dentro do arquivo `GameView.kt`:

12. **Implemente** dentro do **método callback** da interface adicionada no *passo #8* os procedimentos necessários **para obter** o vetor com **a orientação** do dispositivo, e utilize o **primeiro valor do vetor** (*azimuth*) como argumento para chamar o método `atualizarEixo()` (seção 5);

13. Execute os testes no pacote `io.github.vekat.spaceevader (test)` e verifique o resultado.

A.4 Treinamento Gamepad

API da biblioteca Gamepad

Para obtermos acesso aos dados dos sensores de um dispositivo Android por meio da API da biblioteca Gamepad, são necessários pelo menos dois componentes: uma instância configurada da classe `Gamepad`; e uma implementação da interface `ViewHolder`.

Utilização básica

A classe `Gamepad` faz o papel de controle ou *joystick* para o qual serão adicionados um conjunto de botões (classe `Input`) que recebem sinais de diferentes fontes (classe `Source`). Esses objetos são configurados de forma declarativa de forma a facilitar o reúso e a modularidade do tratamento de dados contextuais.

1. Declarando as variáveis config e Gamepad

Funções de extensão são uma das funcionalidades da linguagem Kotlin que foram usadas no *design* da biblioteca **Gamepad**. Para inicializarmos um `Gamepad`, é possível utilizar a função `gamepad()` de uma classe ou objeto que implemente a interface `ViewHolder`. O método `gamepad()` recebe como argumento uma função inicializadora com assinatura `Gamepad.() -> Unit`, ou seja, uma função que possui acesso ao escopo de um `Gamepad` e retorna nada. Todas as alterações realizadas no `Gamepad` desse escopo serão aplicadas ao objeto `Gamepad` que será instanciado e retornado pela função.

Uma vez que funções em Kotlin podem ser armazenadas em variáveis, é possível armazenar a **configuração** de um `Gamepad` em uma variável externa, permitindo a reutilização dessa configuração na criação de novos `Gamepad`s. O exemplo abaixo apresenta como essa tarefa pode ser realizada:

```
// Declarando `config` fora da activity, no escopo do projeto
// permitindo que ela seja reutilizada por outras activities
lateinit var config: Gamepad.() -> Unit

// Exemplo dentro de uma activity chamada `GamepadActivity`
class GamepadActivity : AppCompatActivity() {
  // Declarando um `Gamepad` para agrupar botões
  lateinit var gamepad: Gamepad

  /* ... Restante da Activity ... */
}
```

2. Adicionando a interface ViewHolder

Para recebermos eventos do `Gamepad`, é necessário implementar a interface `ViewHolder`. Essa interface possui uma propriedade do tipo `View` (`instance`), e um método **callback** (`onGamepadEvent()`) que será chamado pela biblioteca sempre que novos eventos forem processados por um `Gamepad`.

Normalmente, essa interface é implementada por uma subclasse de `View` que quer utilizar os dados dos sensores. Para fazer isso, basta adicionar a interface na assinatura da classe, por exemplo:

```
// Adicionando `ViewHolder` na view `ExemploView`
class ExemploView(context: Context) : View(context), ViewHolder {
  // Implementação da propriedade `instance`
  // `instance` aponta para a `View` atual
  override val instance: View = this

  // Implementação vazia do método `onGamepadEvent()`
  override fun onGamepadEvent(event: GamepadEvent) {}

  /* ... Restante da View ... */
}
```

3. Configurando e inicializando o Gamepad

Após a declaração das variáveis mencionadas na seção 1, e a implementação da interface `ViewHolder` em uma *view* como na seção 2, é possível configurar (com a variável `config`) e inicializar o `Gamepad` dentro to método `onCreate()`. É importante notar que a inicialização da variável `config` só precisa ser realizada **uma vez**, pela primeira activity que irá utilizá-la.

```
/* ... Declarações vistas na seção #1 ... */

// Exemplo dentro de uma activity chamada `GamepadActivity`
class GamepadActivity : AppCompatActivity() {
  /* ... Declarações vistas na seção #1 ... */

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    /* ... Outras inicializações relacionadas com a View ... */

    // Inicializando a variável `config` pela primeira vez
```

```
    config = {
      // Declarando um botão do tipo `AxisButton`
      axisButton {
        // Declarando a variável `sensor` como acelerômetro
        val sensor = accelerometer("sensor")

        // Declarando como os dados serão processados
        // pelo botão com a função `handler`
        handler = fun(dados: ContextMap): Float {
          // Recupera os dados mais recentes do sensor
          // acelerômetro e os atribui à variável `acc`
          val acc = dados[sensor]!!

          // Retorna um valor `Float` que é igual à soma
          // das medidas de cada eixo do acelerômetro
          return (acc.x + acc.y + acc.z)
        }
      }
    }

    // Inicializando a variável `gamepad` com a config
    gamepad = view.gamepad(config)
  }
}
```

Diferentes sensores já foram implementados na biblioteca, incluíndo processos mais complexas como obter a orientação do dispositivo. Todos os sensores padrões oferecem funções para facilitar sua criação, como por exemplo o acelerômetro (`accelerometer()`), orientação (`orientation()`) e proximidade (`proximity()`).

4. Ativando e desativando eventos em uma Activity

Após a inicialização do `Gamepad` e dos sensores escolhidos, é necessário chamar seu método `enableInputEvents()` para ativar o recebimento de eventos. Além disso, também é indicado desativar o recebimento de eventos com o método `disableInputEvents()` quando a `Activity` não estiver sendo utilizada. O local ideal para realizar esses procedimentos são os métodos `onResume()` e `onPause()` do ciclo de vida de uma `Activity` , como exemplificado a seguir:

```
// Exemplo dentro de uma activity chamada `SensorActivity`
class SensorActivity : AppCompatActivity() {
```

```
  /* ... Declarações vistas na seção #1 ... */

  /* ... Inicializações vistas na seção #3 ... */

  override fun onResume() {
    super.onResume()

    // Ativa os eventos quando a `Activity` for iniciada
    gamepad.enableInputEvents()
  }

  override fun onPause() {
    super.onPause()

    // Desativa os eventos quando a `Activity` for pausada
    gamepad.disableInputEvents()
  }
}
```

5. Implementando o callback onGamepadEvent

Com a ativação dos eventos, o método **callback** `onGamepadEvent()` passará a receber eventos dos botões configurados na `Activity`. Dessa forma, o último passo é utilizar os dados desses eventos (`event`), que se encontram no vetor `axis`, por exemplo:

```
// Adicionando `ViewHolder` na view `ExemploView`
class ExemploView(context: Context) : View(context), ViewHolder {
  // Implementação da propriedade `instance`
  override val instance = this

  // Implementação simples do método `onGamepadEvent()`
  override fun onGamepadEvent(event: GamepadEvent) {
    // Escreve os valores no vetor do evento
    // para a saída padrão da aplicação
    println( Arrays.toString(event.axis) )
  }

  /* ... Restante da View ... */
}
```

Exemplo avançado

O exemplo a seguir apresenta como podemos configurar um `Gamepad` em uma `Activity` para receber os dados da orientação do dispositivo móvel e enviar apenas a direção do norte magnético (`azimuth`) para uma `View` que implemente a interface `ViewHolder`, permitindo que esse dado seja utilizado de alguma forma:

```
/* Declarando `config` fora da activity (seção 1) */
lateinit var config: Gamepad.() -> Unit

// Exemplo de implementação na activity `OrientationActivity`
class OrientationActivity : AppCompatActivity() {
  // Exemplo de view principal da activity
  lateinit var view: OrientationView

  /* Declarando um `Gamepad` para agrupar botões (seção 1) */
  lateinit var gamepad: Gamepad

  /* No método `onCreate()`, `config` e `gamepad` são inicializados (seção 3) */
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Exemplo de inicialização da view
    view = OrientationView(this)

    setContentView(view)

    // Inicialização da configuração `config`
    config = {
      axisButton {
        val orientacao = orientation("orientacao")

        handler = fun(dados: ContextMap): Float {
          val dadosOrientacao = dados[orientacao]!!

          return dadosOrientacao.azimuth
        }
      }
    }
```

```
    // Inicializando o `gamepad`
    gamepad = view.gamepad(config)
  }

  /* No método `onResume()`, os eventos são ativados (seção 4) */
  override fun onResume() {
    super.onResume()

    // A partir desse momento, é possível receber eventos
    gamepad.enableInputEvents()
  }

  /* No método `onPause()`, os eventos são desativados (seção 4) */
  override fun onPause() {
    super.onPause()

    // A partir desse momento, não é possível receber eventos
    gamepad.disableInputEvents()
  }
}
```

View que implementa a interface `ViewHolder` e utiliza o valor do `azimuth` obtido na activity `OrientationActivity` acima:

```
/* Adicionando `ViewHolder` na view `OrientationView` (seção 2) */
class OrientationView(context: Context) : View(context), ViewHolder {
  // Implementação da propriedade `instance`
  override val instance = this

  /* Implementação do método `onGamepadEvent()` (seção 5) */
  override fun onGamepadEvent(event: GamepadEvent) {
    // Recupera o 'azimuth' na primeira posição
    // do vetor de dados do evento
    val direcaoNorte = event.axis[0]

    /* ... Use `direcaoNorte` de alguma forma ... */
  }
```

```
  /* ... Restante da View ... */
}
```

A.5 Treinamento Android

API padrão do Android SDK

Para acessar os dados dos sensores de um dispositivo Android por meio da API padrão, são necessários pelo menos três componentes: uma instância da classe `SensorManager`; uma ou mais instâncias da classe `Sensor`; e uma implementação da interface `SensorEventListener`.

Utilização básica

Geralmente, a declaração e inicialização da classe `SensorManager` e das instâncias de `Sensor` estão associadas com o ciclo de vida de uma `Activity` Android, enquanto a implementação da interface `SensorEventListener` pode ser realizada de forma independente. Nas seções a seguir, serão exemplificados como isso pode ser implementado.

1. Declarando as variáveis SensorManager e Sensors

Um `SensorManager` pode ser visto como um serviço que pertence ao contexto da aplicação Android (`Context`) e, por esse motivo, a sua inicialização só pode ser realizada *após* termos acesso ao contexto da aplicação. Uma maneira de declarar um `SensorManager` e `Sensor` em Kotlin sem utilizar tipos anuláveis é por meio da palavra chave `lateinit`, por exemplo:

```
// Exemplo dentro de uma activity chamada `SensorActivity`
class SensorActivity : AppCompatActivity() {
  // Declarando um `SensorManager` para administrar os sensores
  lateinit var sensorManager: SensorManager

  // Declarando um `Sensor` para inicializar futuramente
  lateinit var sensor: Sensor

  /* ... Restante da Activity ... */
}
```

2. Adicionando a interface SensorEventListener

A API padrão precisa de uma implementação da interface `SensorEventListener`. Essa interface possui dois métodos **callback** (`onAccuracyChanged()` e `onSensorChanged()`) que serão chamados sempre que novos dados forem capturados pelos sensores registrados. Apenas o método `onSensorChanged()` retorna as leituras dos sensores, sendo o **callback** mais importante para o nosso caso.

Normalmente, essa interface é implementada pela classe que quer utilizar os dados dos sensores, como por exemplo, uma `View`. Para fazer isso, basta adicionar a interface na assinatura da classe, por exemplo:

```
// Adicionando `SensorEventListener` na view `ExemploView`
class ExemploView(context: Context): View(context), SensorEventListener {
  // Implementação vazia do método `onSensorChanged()`
  override fun onSensorChanged(event: SensorEvent?) {}

  // Implementação vazia do método `onAccuracyChanged()`
  override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}

  /* ... Restante da View ... */
}
```

3. Inicializando o SensorManager e Sensors

Como mencionado na seção 1, somente é possível obter instâncias de `SensorManager` e `Sensor` quando o contexto da aplicação for devidamente inicializado. Dessa forma, a inicialização das variáveis declaradas na primeira seção desse documento é realizada no método `onCreate()` da seguinte forma:

```
// Exemplo dentro de uma activity chamada `SensorActivity`
class SensorActivity : AppCompatActivity() {
  /* ... Declarações vistas na seção #1 ... */

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    /* ... Outras inicializações relaciondas com a View ... */

    // Inicializando a variável `sensorManager`
    sensorManager = getSystemService(Context.SENSOR_SERVICE) as SensorManager

    // Inicializando a variável `sensor` como acelerômetro
    sensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
  }
}
```

Para cada dispositivo Android, existem diferentes tipos e quantidades de sensores disponíveis. No entanto, os sensores **acelerômetro** (`Sensor.TYPE_ACCELEROMETER`) e **magnetômetro** (`Sensor.TYPE_MAGNETIC_FIELD`) estão presentes na maioria dos dispositivos.

4. Ativando e desativando eventos em uma Activity

Após a inicialização do `SensorManager` e dos sensores escolhidos, é necessário chamar o método `registerListener()` para ativar o recebimento de eventos. Além disso, também é indicado desativar o recebimento de eventos com o método `unregisterListener()` quando a `Activity` não estiver sendo utilizada. O local ideal para realizar esses procedimentos são os métodos `onResume()` e `onPause()` do ciclo de vida de uma `Activity`, como exemplificado a seguir:

```
// Exemplo dentro de uma activity chamada `SensorActivity`
class SensorActivity : AppCompatActivity() {
  /* ... Declarações vistas na seção #1 ... */

  /* ... Inicializações vistas na seção #3 ... */

  override fun onResume() {
    super.onResume()
    // Registrando o listener `view` para o `sensor`
    // com velocidade de eventos `SENSOR_DELAY_GAME`
    sensorManager.registerListener(view, sensor, SensorManager.SENSOR_DELAY_GAME)
  }

  override fun onPause() {
    super.onPause()
    // Desativando o listener `view` para todos
    // os sensores registrados anteriormente
    sensorManager.unregisterListener(view)
  }
}
```

5. Implementando o callback onSensorChanged

Com a `Activity` devidamente configurada, o método `onSensorChanged()` passará a receber eventos de todos os sensores registrados para essa instância. Dessa forma, o último passo é utilizar os dados do evento (`event`), que se encontram no vetor `values`, por exemplo:

```
// Adicionando `SensorEventListener` na view `ExemploView`
class ExemploView(context: Context): View(context), SensorEventListener {
  // Implementação simples do método `onSensorChanged()`
  override fun onSensorChanged(event: SensorEvent?) {
```

```
    // Verifica se o evento não é nulo
    if (event != null) {
      // Escreve os valores no vetor do evento
      // para a saída padrão da aplicação
      println( Arrays.toString(event.values) )
    }
  }

  // Implementação vazia do método `onAccuracyChanged()`
  override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}

  /* ... Restante da View ... */
}
```

Exemplo avançado

O sensor **acelerômetro** mede a aceleração aplicada ao dispositivo (m/s*s). Já o sensor **magnetômetro** mede os campos magnéticos do ambiente. Ambos os sensores retornam 3 valores, correspondendo aos eixos X, Y e Z. É possível combinar os dados medidos pelos dois sensores para obter a **orientação** do dispositivo móvel, por meio dos métodos `SensorManager.getOrientation()` e `SensorManager.getRotationMatrix()` da seguinte forma:

```
// Exemplo de implementação na activity `OrientationActivity`
class OrientationActivity : AppCompatActivity() {
  // Exemplo de view principal da activity
  lateinit var view: OrientationView

  /* Declarações das variáveis necessárias (seção 1) */
  // Um `SensorManager` para administrar os sensores
  lateinit var sensorManager: SensorManager
  // Um `Sensor` para o acelerômetro
  lateinit var sAcc: Sensor
  // Um `Sensor` para o magnetômetro
  lateinit var sMag: Sensor

  /* No método `onCreate()`, as variáveis são inicializadas (seção 3) */
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    // Exemplo de inicialização da view
```

```
    view = OrientationView(this)

    setContentView(view)

    // Inicialização do `sensorManager`
    sensorManager = getSystemService(Context.SENSOR_SERVICE)

    // Inicialização dos sensores
    sAcc = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
    sMag = sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)
  }

  /* No método `onResume()`, os eventos são ativados (seção 4) */
  override fun onResume() {
    super.onResume()

    // A partir desse momento, é possível receber eventos
    sensorManager.registerListener(view, sAcc, SensorManager.SENSOR_DELAY_GAME)
    sensorManager.registerListener(view, sMag, SensorManager.SENSOR_DELAY_GAME)
  }

  /* No método `onPause()`, os eventos são desativados (seção 4) */
  override fun onPause() {
    super.onPause()

    // A partir desse momento, não é possível receber eventos
    sensorManager.unregisterListener(view)
  }
}
```

View que implementa a interface `SensorEventListener` e processa os valores do acelerômetro e magnetômetro:

```
/* Adicionando `SensorEventListener` na view `OrientationView` (seção 2) */
class OrientationView(context: Context) : View(context), SensorEventListener {
  // Declare dois vetores para guardar os valores dos sensores
  var dadosAcc: FloatArray? = null
  var dadosMag: FloatArray? = null

  /* Implementação do método `onSensorChanged()` (seção 5) */
```

```
  override fun onSensorChanged(event: SensorEvent?) {
    if (event != null) {
      // Descobre qual sensor enviou dados e
      // salva os dados no vetor correto
      when (event.sensor.type) {
        Sensor.TYPE_ACCELEROMETER -> dadosAcc = event.values
        Sensor.TYPE_MAGNETIC_FIELD -> dadosMag = event.values
      }

      // Se os dois vetores possuírem dados
      if (dadosAcc != null && dadosMag != null) {
        // Declara um vetor para representar a matriz
        // rotacional do dispositivo que será calculada
        val mRot = FloatArray(9)

        // Calcula a matriz rotacional do dispositivo
        // a partir dos dados dos sensores
        if (SensorManager.getRotationMatrix(mRot, null, dadosAcc, dadosMag)) {
          val orientacao = FloatArray(3)

          // Calcula o vetor de orientação do dispositivo
          // em ângulos de Euler
          SensorManager.getOrientation(mRot, orientacao)

          // O primeiro valor do vetor de orientação
          // indica o 'azimuth', a direção do Norte magnético
          val direcaoNorte = orientacao[0]

          /* ... Use `direcaoNorte` de alguma forma ... */
        }
      }
    }
  }

  override fun onAccuracyChanged(sensor: Sensor?, precisao: Int) {}
}
```

APÊNDICE B – Materiais da Análise Estatística

B.1 Implementação do teste de normalidade

```r
library(sm)

pdf("density.pdf")
df <- read.csv("dat-exp.csv")
size = 1.3
# classe dos tratamentos
trea <- c(1,1,1,1,1,2,2,2,2,2)
trea.f <- factor(trea, levels=c(1,2), labels=c("Gamepad", "Padrão"))

lloc <- c(df$lloc_gamepad, df$lloc_base)
mcc <- c(df$mcc_gamepad, df$mcc_base)
time <- c(df$time_gamepad, df$time_base)

# criação dos gráficos de densidade
sm.density.compare(lloc, trea, xlab="Linhas de Código-Fonte", ylab="Densidade",
 ↪  cex=size)
title(main="Distribuição de Tamanho por Tratamento", cex=size)
legend("topright", levels(trea.f), fill=rainbow(2), cex=size)

sm.density.compare(mcc, trea, xlab="Complexidade de McCabe", ylab="Densidade", cex=size)
title(main="Distribuição de Complexidade por Tratamento", cex=size)
legend("topright", levels(trea.f), fill=rainbow(2), cex=size)

sm.density.compare(time, trea, xlab="Duração em Segundos", ylab="Densidade", cex=size)
title(main="Distribuição do Tempo por Tratamento", cex=size)
legend("topright", levels(trea.f), fill=rainbow(2), cex=size)

dev.off()

# execução do teste shapiro-wilk
shapiro.test(df$lloc_gamepad)
shapiro.test(df$lloc_base)

shapiro.test(df$mcc_gamepad)
shapiro.test(df$mcc_base)

shapiro.test(df$time_gamepad)
shapiro.test(df$time_base)
```

Listagem 8 – Código R com os parâmetros usados no teste Shapiro-Wilk.

B.2 Implementação do teste das hipóteses

```r
df = read.csv("dat-exp.csv")

# nível de confiança = 90%
CONF_LVL = 0.9
# número de amostras por par = 5
NUM = 5
# tipo: A != B
TYPE = "two.sided"
# tratamentos: com e sem biblioteca
TREATMENTS = rep(c("Gamepad", "Padrão"), each=NUM)

# criação das distribuições
lloc_data <- data.frame(metric = c(df$lloc_gamepad, df$lloc_base), treatment = TREATMENTS)

mcc_data <- data.frame(metric = c(df$mcc_gamepad, df$mcc_base), treatment = TREATMENTS)

time_data <- data.frame(metric = c(df$time_gamepad, df$time_base), treatment = TREATMENTS)

# execução do teste wilcoxon
lloc_res <- wilcox.test(metric ~ treatment, data = lloc_data, alternative = TYPE, conf.int = TRUE, conf.lev = CONF_LVL, paired = TRUE)

mcc_res <- wilcox.test(metric ~ treatment, data = mcc_data, alternative = TYPE, conf.int = TRUE, conf.lev = CONF_LVL, paired = TRUE)

# execução do teste t
timet_res <- t.test(metric ~ treatment, data = time_data, alternative = TYPE, paired = TRUE)

# função para obter o 'effect size'
rwilcox <- function(wilcoxRes, n) {
  z <- qnorm(wilcoxRes$p.value / 2)
  r <- z / sqrt(n)
  cat("effect size:\t", wilcoxRes$data.name, "\nr = ", r, ", z = ", z, "\n")
}

# exibição dos resultados
lloc_res
rwilcox(lloc_res, NUM)
mcc_res
rwilcox(mcc_res, NUM)
timet_res
```

Listagem 9 – Código R com os parâmetros usados no teste Wilcoxon.

APÊNDICE C – Proposta de Experimento

Plano experimental

Avaliação da biblioteca Gamepad

1 Escopo do estudo

1.1 Objetivo

Esse estudo experimental tem como **objetivo global** avaliar se a biblioteca proposta pelo presente trabalho proporciona alguma *melhoria* ao processo de manipulação dos dados contextuais no desenvolvimento de jogos pervasivos móveis.

Dessa forma, esse estudo experimental busca **analisar** o processo de tratamento do contexto nos jogos pervasivos móveis *com* e *sem* a utilização da biblioteca proposta, com o **propósito** de avaliar as soluções implementadas **com respeito** ao esforço empregado na implementação e a qualidade da solução implementada no **ponto de vista** do desenvolvedor e pesquisador de jogos pervasivos móveis, dentro do **contexto** acadêmico com estudantes de graduação e pós-graduação implementando jogos pervasivos móveis.

1.2 Questões da pesquisa

Com base no objetivo definido, as principais questões de pesquisa que norteiam esse estudo são apresentadas a seguir:

- **Questão 1**: O esforço exigido pelas implementações que utilizaram a biblioteca proposta é inferior ao esforço requerido sem a utilização da biblioteca?
- **Questão 2**: A extensão das soluções que utilizaram a biblioteca proposta é inferior à extensão das soluções que não utilizaram a biblioteca?
- **Questão 3**: A complexidade do código-fonte produzido com a biblioteca proposta é menor do que a complexidade do código-fonte produzido sem a utilização da biblioteca?
- **Questão 4**: A percepção de facilidade na implementação com o apoio da biblioteca proposta é melhor do que a percepção de facilidade sem a biblioteca?

2 Planejamento do estudo

2.1 Definição do contexto

O experimento deverá ser realizado em um ambiente laboratorial, ***in vitro***, de forma controlada, o que caracteriza uma execução ***offline***. Os participantes

(ou sujeitos) do experimento serão primariamente **estudantes universitários** de graduação e pós-graduação.

Dois projetos distintos de aplicativo para a plataforma Android serão utilizados como objeto pelos participantes Os projetos estarão parcialmente implementados, contendo um conjunto limitado de **problemas modelados** que deverão ser solucionados pelos participantes. Essa escolha tem um impacto no desenho experimental, e será discutida futuramente.

O experimento é de caráter **específico**, pois este visa comparar uma abordagem existente com uma abordagem alternativa. Esse tipo de comparação possui implicações que podem afetar o comportamento do participante em relação às abordagens testadas, e será discutido na seção de ameaças a validade.

2.2 Formulação das hipóteses

Na apresentação do escopo foi expressado que o objetivo do estudo é comparar as soluções que usam a biblioteca proposta (CB) e as que não usam a biblioteca (SB) com respeito às variáveis *esforço*, *extensão*, *complexidade* e *qualidade*. Essas variáveis serão devidamente detalhadas na subseção 2.3.

Dessa forma, sejam:

- $\mu_{e_{CB}}$ e $\mu_{e_{SB}}$ as médias dos esforços empregados pelos participantes usando as abordagens CB e SB, respectivamente;
- $\mu_{x_{CB}}$ e $\mu_{x_{SB}}$ as médias das extensões dos códigos-fonte implementados usando as abordagens CB e SB, respectivamente;
- $\mu_{c_{CB}}$ e $\mu_{c_{SB}}$ as médias das complexidades dos códigos-fonte implementados usando as abordagens CB e SB, respectivamente; e
- $\mu_{q_{CB}}$ e $\mu_{q_{SB}}$ as médias das qualidades percebidas pelos participantes usando as abordagens CB e SB, respectivamente.

As seguintes hipóteses foram formuladas para cada variável:

- Esforço
  Hipótese nula ($H0_e$): $\mu_{e_{CB}} = \mu_{e_{SB}}$, isto é, o *esforço* exigido nas implementações *com a biblioteca* proposta *não é* significantemente *diferente do esforço* exigido nas implementações *sem a biblioteca*;
  Hipótese alternativa ($H1_e$): $\mu_{e_{CB}} \neq \mu_{e_{SB}}$;
- Extensão
  Hipótese nula ($H0_x$): $\mu_{x_{CB}} = \mu_{x_{SB}}$, isto é, a *extensão* observada nas implementações *com a biblioteca* proposta *não é* significantemente *diferente da extensão* observada nas implementações *sem a biblioteca*;
  Hipótese alternativa ($H1_x$): $\mu_{x_{CB}} \neq \mu_{x_{SB}}$;
- Complexidade
  Hipótese nula ($H0_c$): $\mu_{c_{CB}} = \mu_{c_{SB}}$, isto é, a *complexidade* observada nas implementações *com a biblioteca* proposta *não é* significantemente *diferente*

da complexidade observada nas implementações *sem a biblioteca*;
Hipótese alternativa ($H1_c$): $\mu_{c_{CB}} \neq \mu_{c_{SB}}$;

- Qualidade
  Hipótese nula ($H0_q$): $\mu_{q_{CB}} = \mu_{q_{SB}}$, isto é, a *qualidade* percebida nas soluções implementadas *com a biblioteca* proposta *não é* significantemente *diferente da qualidade* percebida nas soluções implementadas *sem a biblioteca*;
  Hipótese alternativa ($H1_q$): $\mu_{q_{CB}} \neq \mu_{q_{SB}}$.

2.3 Definição das variáveis

- **Variáveis independentes**: parâmetros que causam os resultados observados.
  1. Linguagem de programação e a plataforma selecionada: Android e Kotlin;
  2. Experiência dos participantes com as ferramentas escolhidas;
  3. Ferramenta de desenvolvimento para lidar com dados contextuais: (*i*) SDK da plataforma e a biblioteca de controladores *fuzzy* jFuzzyLite (*baseline*); (*ii*) Biblioteca proposta por este trabalho.

- **Variáveis dependentes**: refletem o efeito das variáveis independentes.
  1. Tempo gasto para concluir a implementação da tarefa;
  2. Quantidade de linhas executáveis no código fonte implementado;
  3. Complexidade condicional do código fonte implementado;
  4. Percepção de qualidade da implementação e compreensão da solução;

2.4 Seleção dos participantes

Antes da realização do estudo, os candidatos a participarem do experimento serão caracterizados por meio de um questionário de recrutamento. Os participantes selecionados para esse estudo serão alunos de graduação e pós-graduação do curso de Ciência da Computação. É um requisito que os participantes tenham disponibilidade para comparecer à execução do estudo e que possuam conhecimentos suficientes sobre o desenvolvimento de jogos e aplicativos para a plataforma Android.

O formulário de recrutamento deverá requisitar do candidato alguns dados de identificação e informações sobre sua disponibilidade de tempo para a participação no experimento. Além disso, os candidatos também devem responder questões sobre as tecnologias que serão utilizadas no experimento, apresentadas a seguir:

1. **Q:** Em qual setor você atua?
   R: Múltipla escolha: (*i*) setor acadêmico (ensino); (*ii*) estudante (graduação ou pós-graduação); (*iii*) setor industrial (empresarial).

2. **Q:** Você tem quantos anos de experiência com desenvolvimento Android?
   R: Múltipla escolha: (*i*) menos do que 1 ano; (*ii*) cerca de 1 a 2 anos; (*iii*) cerca de 3 a 4 anos; (*iv*) mais do que 5 anos.

3. **Q:** Você já usou a linguagem Kotlin?
   R: Sim ou não.

2.5 Desenho experimental

O desenho experimental define como os tratamentos serão aplicados aos objetos do estudo. Além dos tratamentos escolhidos, os participantes também serão requisitados a responder dois questionários, um antes e depois da tarefa do experimento. Esse estudo adota um desenho experimental com **um fator** e **dois tratamentos** sem *crossover*, apresentado na Tabela 1.

Tabela 1: Desenho experimental do estudo proposto.

	Implementação dos objetos do estudo			
	Etapa 1		**Etapa 2**	
Grupo	Grupo de controle		Grupo de experimento	
Tratamentos	Treinamento	Tarefas sem a biblioteca	Treinamento	Tarefas com a biblioteca

Neste tipo de estudo, os participantes são divididos em dois grupos, um grupo de controle e um grupo experimental, e o estudo é dividido em duas etapas intermutáveis, uma para cada grupo. É importante que as etapas sejam idênticas com respeito aos tratamentos, materiais e instruções apresentadas aos participantes, com exceção da ferramenta que pretende-se verificar no experimento, ou seja, o grupo de controle deve realizar as mesmas tarefas do grupo de experimento, porém sem utilizar a biblioteca proposta. Dessa forma, caso exista uma diferença significante entre os resultados dos grupos, é possível afirmar que tal diferença foi causada pela ferramenta escolhida.

Para que seja possível garantir a aleatoriedade dos grupos formados e evitar que um grupo tenha vantagem sobre o outro, pretende-se bloquear (*blocking*) o nível de experiência dos participante com a linguagem de programação e a plataforma utilizada. Os parâmetros para dividir os blocos serão obtidos do formulário de recrutamento mencionado na subseção anterior. Por fim, os testes estatísticos aplicáveis a esse desenho são os paramétricos *t-test* e *F-test* ou alternativamente os não-paramétricos *Mann-Whitney* e *Chi-2*.

2.6 Instrumentação

Esse estudo utiliza dois objetos no qual serão aplicados os tratamentos pelos participantes, apresentados a seguir:

- **Uso dos dados contextuais**: Implementar os procedimentos para realizar a coleta, pré-processamento e utilização dos dados que fazem parte do contexto, cujo acesso é proporcionado pelos sensores embutidos dos dispositivos móveis. Essa tarefa prática envolve adicionar ao jogo base a funcionalidade de responder a informações contextuais para controlar o jogo;

- **Tratamento das incertezas**: Implementar os procedimentos para realizar a modelagem, pré-processamento e utilização dos dados contextuais com a presença de incertezas, por exemplo, baixa precisão em medições, graus de confiança, possibilidade de desconexão. Essa tarefa prática envolve aplicar a

teoria de conjuntos *fuzzy* para representar a incerteza dos dados sensoreados e utilizar um controlador *fuzzy* para determinar o comportamento do jogo.

Cada objeto de estudo será acompanhado por um projeto Android pré-configurado com o ponto de partida da solução, um conjunto de instruções que descrevem como alcançar o objetivo final da solução explicitando as funcionalidades esperadas.

2.7 Ameaças à validade

- **Validade de conclusão**: Foram consideradas as seguintes ameaças: (*i*) **escolher testes com baixo poder estatístico**: espera-se evitar essa ameaça com a seleção de testes estatísticos voltados para o desenho experimental escolhido e o tipo das variáveis dependentes; (*ii*) **violar premissas dos testes estatísticos**: espera-se evitar essa ameaça com a realização de uma análise prévia dos dados e a possível eliminação de *outliers*; (*iii*) **escolher medições com baixa confiabilidade**: espera-se mitigar essa ameaça com a utilização de medidas objetivas que não dependem de interpretação, nesse estudo serão adotadas quatro medidas objetivas e uma subjetiva; (*iv*) **aplicar tratamentos com alta variabilidade**: espera-se mitigar essa ameaça com a aplicação de dois tratamentos com uma única implementação, para que não exista a possibilidade de aplicar o mesmo tratamento de forma diferente para os participantes.

- **Validade interna**: Para fortalecer o relacionamento de causa-efeito entre os tratamentos e os resultados, foram consideradas as seguintes ameaças: (*i*) **agrupar participantes de forma inadequada**: espera-se mitigar essa ameaça aplicando um balanceamento do nível de experiência dos participantes com as tecnologias utilizadas baseado em informações obtidas por meio de questionários de recrutamento; (*ii*) **maturação**: seguindo o desenho experimental proposto, os tratamentos não deverão ser repetidos mais de uma vez para o grupo de experimento.

- **Validade de construção**: Representa o grau de relacionamento entre teoria e observação, foram consideradas as seguintes ameaças: (*i*) **confiabilidade do desenho experimental**: visando um *trade-off* entre custo e precisão dos resultados, o desenho experimental escolhido não realiza um *crossover* dos participantes e em troca, utiliza um *control group*; (*ii*) **expectativa dos pesquisadores**: espera-se mitigar essa ameaça com a proibição de qualquer ajuda por parte dos pesquisadores durante a execução das etapas de implementação do experimento.

- **Validade externa**: Foram consideradas as seguintes ameaças: (*i*) **selecionar participantes inadequados**: espera-se mitigar essa ameaça aplicando um questionário para o recrutamento de participantes com experiência nos temas abordados e tecnologias utilizadas, entre alunos de graduação, pós-graduação e profissionais; (*ii*) **selecionar ferramentas inadequadas**: a seleção de ferramentas empregadas pelo grupo de controle serão verificadas por estudos e estatísticas que comprovem sua utilização na prática industrial e acadêmica.