UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

MESTRADO E DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

EVILASIO COSTA JUNIOR

MOTION - PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES DE *INTERNET OF HEALTH THINGS* AUTOADAPTATIVAS BASEADAS EM PADRÕES DE MOVIMENTO

FORTALEZA

2023

EVILASIO COSTA JUNIOR

MOTION - PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES DE *INTERNET OF HEALTH THINGS* AUTOADAPTATIVAS BASEADAS EM PADRÕES DE MOVIMENTO

Tese apresentada ao Mestrado e Doutorado em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Engenharia de Software

Orientadora: Prof. Dra. Rossana Maria de Castro Andrade

Coorientador: Prof. Dr. Leonardo Sampaio Rocha

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C871m Costa Junior, Evilasio.
MOTION - Processo de desenvolvimento de aplicações de Internet of Health Things autoadaptativas baseadas em padrões de movimento / Evilasio Costa Junior. – 2023.
186 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2023.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.
Coorientação: Prof. Dr. Leonardo Sampaio Rocha.

1. processo de software. 2. internet das coisas médicas. 3. sistemas autoadaptativos. 4. saúde digital. 5. reúso de software. I. Título.

CDD 005

EVILASIO COSTA JUNIOR

MOTION - PROCESSO DE DESENVOLVIMENTO DE APLICAÇÕES DE *INTERNET OF HEALTH THINGS* AUTOADAPTATIVAS BASEADAS EM PADRÕES DE MOVIMENTO

Tese apresentada ao Mestrado e Doutorado em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Engenharia de Software

Aprovada em: 28/08/2023

BANCA EXAMINADORA

Prof. Dra. Rossana Maria de Castro Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Leonardo Sampaio Rocha (Coorientador)
Universidade Estadual do Ceará (UECE)

Prof. Dra. Carla Andrea Taramasco Toro
Universidad Andrés Bello (UNAB)

Prof. Dr. Danielo Gonçalves Gomes
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Lopes Gomes
Universidade Estadual do Ceará (UECE)

Prof. Dr. Miguel Franklin de Castro
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe e Pai, seus cuidados e dedicação deram a esperança para seguir, e suas presenças significaram segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

A Profa. Dra. Rossana Maria de Castro Andrade por me orientar em minha tese de doutorado.

Ao Prof. Dr. Leonardo Sampaio Rocha, pela coorientação em minha tese de doutorado.

Aos meus pais, irmãos e sobrinho, que nos momentos de minha ausência dedicados ao estudo superior, sempre me apoiaram e me ajudaram a continuar focado no meu objetivo.

Aos meus amigos e amigas que ouviram meus lamentos e alegrias e que me apoiaram nesse período tão marcante da minha vida. Obrigado Davi, Lucas, Raphaella, Carlírio, Joana, Yuri, Amanda, Rodrigo, Gabriel, Edson, Erick, Elisabete, Antônio, Ian, Bruno, Wesllei, Giulliano, Aldemir, Michael, Melissa, Henrique, Guilherme, Nayara, Luiz e aos demais amigos com quem convivi durante esses últimos seis anos.

Aos meus amigos e colegas de laboratório, pelas discussões sobre a pesquisa, apoio nos estudos, conversas durante o cafezinho e por todo apoio que prestaram durante essa jornada. Muito Obrigado pelo apoio Pedro, Italo, Belmondo, Amanda, Rainara, Ismayle, Ticiane, Josy, Tales, Bruno, Rute, Armando, Manoel, Maurício, Almada, Rhenara, Laurindo, Cleitianne, Vitor, Nayana, Lourival, Breno, Mario, Anne, Adriano e todos os demais que não consegui citar.

A todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), na pessoa da Presidente Mercedes Bustamante pelo financiamento da pesquisa de doutorado via bolsa de estudos.

E ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ), na pessoa do Presidente Ricardo Magnus Osório Galvão pelo financiamento da pesquisa de doutorado via bolsa, pela participação no projeto INCT Ines 2.0.

“Nunca deixe seu senso moral impedir você de fazer o que é certo!”

(Isaac Assimov)

RESUMO

Informações sobre o movimento de uma pessoa, como marcha, velocidade, postura e localização, podem indicar a existência de problemas de saúde, principalmente em idosos. Para monitorar esses padrões de movimento podem ser usados sistemas de *Internet of Things* (IoT), que são sistemas computacionais que usam objetos inteligentes conectados à internet e munidos de sensores para monitorar dados no ambiente. Por meio de dispositivos IoT, é possível identificar padrões de movimento e utilizá-los para monitorar o estado de saúde de uma pessoa. O termo *Internet of Health Things* (IoHT) vem sendo utilizado na literatura para identificar soluções IoT para a saúde e diversos estudos propõem sistemas IoHT baseados em dados de movimento. No entanto, ainda há desafios para construção destes sistemas, como a falta de um processo de desenvolvimento de *software* específico. Além disso, outros dois desafios em aberto são: como relacionar os dados dos sensores com os problemas de saúde e como tornar esses sistemas menos intrusivos e mais ubíquos, de modo a acionar os usuários apenas quando necessário. Uma solução para criação de sistemas ubíquos e pouco intrusivos pode ser a construção de sistemas IoHT autoadaptativos. Considerando isso, nessa tese é proposto um processo de desenvolvimento de *software* para aplicações IoHT autoadaptativas baseadas em dados de movimento, intitulado MOTION. Também são propostos três artefatos de reúso para auxiliar o desenvolvimento de aplicações IoHT, sendo eles: (i) um grafo de classificação, que relaciona sensores e situações de saúde para auxiliar a elicitação de requisitos e o projeto da aplicação IoHT, podendo também ser usado com base de conhecimento pela aplicação; (ii) um *template* para auxiliar a construção de regras de adaptação; e (iii) um *framework* em Kotlin para auxiliar a implementação de aplicações IoHT autoadaptativas para dispositivos Android. O processo MOTION foi avaliado através de um experimento feito em conjunto com profissionais de Tecnologia de Informação e Comunicação (TIC), que utilizaram o processo para auxiliar o desenvolvimento de duas aplicações IoHT autoadaptativas baseadas em dados de movimento. Já os artefatos de reúso foram avaliados por meio de provas de conceito.

Palavras-chave: processo de *software*; internet das coisas médicas; sistemas autoadaptativos; saúde digital; reúso de *software*.

ABSTRACT

Information about a person's movement, such as gait, speed, posture, and location, can indicate health problems, particularly in older adults. These movement patterns can be monitored using *Internet of Things* (IoT) systems, which are computational systems that use smart objects connected to the internet and equipped with sensors to monitor data in the environment. Using IoT devices, it is possible to identify movement patterns and use them to monitor a person's health status. The term *Internet of Health Things* (IoHT) has been used in the literature to identify IoT solutions for health and several studies have already proposed IoHT systems based on movement data. However, there are still challenges in building these systems, such as the lack of a specific software development process. In addition, two other open challenges are: how to relate sensor data to health problems and how these systems can be less intrusive and more ubiquitous to trigger users, only when necessary. A solution for creating ubiquitous and less intrusive systems can be the construction of self-adaptive IoHT systems. Considering this, this thesis proposes a software development process for self-adaptive IoHT applications based on motion data, entitled MOTION. Three reuse artifacts are also proposed to aid the development of IoHT applications. They are: (i) a classification graph, which relates sensors and health situations to help elicit requirements and design the IoHT application, which can also be used as a knowledge base by the application; (ii) a *template* to help build adaptation rules; and (iii) a *framework* in Kotlin to help implement self-adaptive IoT applications for Android devices. The MOTION process is evaluated through an experiment with Information and Communication Technology professionals, who used the process to help the development of two self-adaptive IoHT applications based on motion data. Moreover, the reuse artifacts were evaluated through a proof of concept.

Keywords: software process; internet of health things; self-Aaaptive systems; digital health; software reuse.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAL	*Ambient Assisted Living*
ADL	*Activity Daily Living*
ARCANA	*Activity Recognition and Cardiology ANALlysis*
ARTe	*Adaptation Rules Template*
AWS	*Amazon Web Services*
COTS	*Commercial-off-the-shelf*
DA	*Discriminant Analysis*
DAO	*Data Access Object*
DCA	*Deep Collaborative Alert Recommendation*
FAD	Facilidade de uso do processo para desenvolvimento de um sistema IoHT autoadaptativo
FAP	Facilidade de Aderência ao Processo
GNB	*Gaussian Naive Bayes*
GRAFIT	*Classification Graph for IoHT Applications*
HMM	*Hidden Markov Models*
IoHT	*Internet of Health Things*
IoT	*Internet of Things*
KNN	*K-nearest neighbors*
KREATION	*Kotlin Framework for Self-adaptive IoHT Applications*
LCSS	*Longest Common SubSequence*
MAE	*Mean Absolute Error*
MD	*Manhattan Distance*
MOTION	Processo de Desenvolvimento para Aplicações IoHT Autoadaptativas Baseadas em Padrões de Movimento
MVC	*Model View Control*
NDCG	*Normalized Discounted Cumulative Gain*
OIS	Objetos Inteligentes de Saúde
OMS	Organização Mundial da Saúde
PCA	*Principal Component Analysis*
PCC	*Pearson's Correlation Coefficient*
PoC	*Proof of Concept*

PUM	Percepção de utilidade do manual de uso e execução do processo MOTION
PUP	Percepção de utilidade do processo MOTION
REALER	*Risk movEment ALERt*
RMSE	*Root-Mean-Square Error*
SAS	*Self-Adaptive System*
SVM	*Support Vector Machines*
TAM	*Technology Acceptance Model*
TAR	*Technical Action Research*
TCLE	Termo de Consentimento Livre Esclarecido
TIC	Tecnologia de Informação e Comunicação
TUG	teste *Time Up & Go*

SUMÁRIO

1 INTRODUÇÃO

Este trabalho de doutorado propõe o MOTION, um processo de desenvolvimento de *software* para sistemas autoadaptativos de *Internet of Health Things* (IoHT) baseados em padrões de movimentação. Além do processo de desenvolvimento, também são propostos três artefatos de *software* para auxiliar a execução das atividades de desenvolvimento de *software* do processo MOTION, a seguir: um grafo de classificação, intitulado GRAFIT, um *template* para definição das regras de adaptação, intitulado ARTe, e um *framework* em Kotlin, intitulado KREATION, para auxiliar a implementação de aplicações IoHT autoadaptativos para dispositivos Android.

Este capítulo apresenta a introdução deste trabalho, começando pela contextualização na seção 1.1; na seção 1.2 é apontada a sua motivação; a seção 1.3 contém a hipótese, o objetivo e as questões de pesquisa que guiam este trabalho; a estratégia usada nesta pesquisa é apresentada na seção 1.4; por último, a seção 1.5 apresenta a estrutura deste documento.

1.1 Contextualização

O envelhecimento da população mundial é um fenômeno que vem afetando cada vez mais países nas últimas décadas (ROBBINS *et al.*, 2018). Esse fenômeno implica em um aumento da demanda de soluções que permitam melhorar a saúde de idosos, de modo a prover mais independência aos mesmos. Além disso, segundo a Organização Mundial da Saúde (OMS) (WHO, 2016) criar novas aplicações voltadas a melhorar a saúde de idosos é uma ação estratégica fundamental para lidar com o fenômeno do envelhecimento da população mundial. Com isso, é possível perceber um aumento recente de interesse tanto da academia quanto do mercado pela proposição e desenvolvimento de soluções tecnológicas e computacionais para suporte a saúde (ZEADALLY *et al.*, 2020).

Neste sentido, a análise de padrões de movimento possibilita a descoberta de indicativos de problemas de saúde (KIRKWOOD *et al.*, 2018). A marcha, a postura e a velocidade da caminhada podem, por exemplo, indicar dor, riscos de queda, doenças crônicas e problemas estruturais em ossos (PEIMANKAR *et al.*, 2023; MAGNO *et al.*, 2018; STUDENSKI, 2009; MOCK *et al.*, 2007). Outra informação que pode ser analisada é a localização. Com essa informação é possível identificar, por exemplo, a noctúria (TARAMASCO *et al.*, 2018), que é a necessidade de urinar várias vezes a noite, o que pode ter causas cardíacas, hepáticas, autoimunes ou renais.

Esses padrões de movimento podem ser monitorados de maneira automática com o uso de sistemas computacionais, como os sistemas de Internet das Coisas. A Internet das Coisas, do inglês *Internet of Things* (IoT), é um conceito que propõe uma infraestrutura onde objetos do cotidiano identificáveis e dotados de algum tipo processamento se integram através da internet, a uma rede global, dinâmica, autoconfigurável e interoperável (SUNDMAEKER *et al.*, 2020). Esses objetos possuem sensores e atuadores que permitem monitorar informações diversas no ambiente e interagem de modo a influenciar o mundo físico.

Inúmeros sistemas vem utilizando dispositivos de IoT para monitorar e prover melhorias a saúde de pessoas (GUBBI *et al.*, 2013; CHATTERJEE; ARMENTANO, 2015). Esses são os chamados sistemas de *Internet of Health Things* (IoHT), terminologia utilizada nos últimos anos para se referir a soluções IoT voltadas ao monitoramento e melhoria de saúde (RODRIGUES *et al.*, 2018).

Vários trabalhos na última década exploram o uso de IoHT em conjunto com padrões de movimento (KASHANI *et al.*, 2021). Em (LI *et al.*, 2022), (LINHARES *et al.*, 2020) e (MAKHLOUF *et al.*, 2018) são apresentadas soluções IoHT para detecção automática de quedas. Já (HOSSAIN; INOUE, 2019) e (PALUMBO *et al.*, 2017) propõem abordagens usando soluções IoHT para monitoramento e identificação de atividades diárias e situações anômalas. Em (PARK *et al.*, 2020) são utilizados sensores de pressão e aceleração e dispositivos vestíveis para identificar se a pessoa já passou por um acidente vascular cerebral.

Já os sistemas ubíquos (AL-FUQAHA *et al.*, 2015), que são uma das principais bases da IoT, representam sistemas capazes de disponibilizar serviços diversos aos usuários a qualquer momento e em qualquer lugar, de modo que os sistemas apenas interajam com o usuário quando necessário. Assim sendo, na maior parte do tempo esses sistemas executam funções transparentes para os usuários. Nesse sentido, surgem os sistemas autoadaptativos que são capazes de se adaptar automaticamente em resposta a estímulos, necessitando o mínimo possível de intervenção da pessoa que está utilizando o sistema (BRUN *et al.*, 2009). Sistemas autoadaptativos são utilizados para construção de diversas soluções IoT (IFTIKHAR *et al.*, 2017; BURGER *et al.*, 2020; KRUPITZER *et al.*, 2020), no entanto, ainda são pouco utilizados para IoHT.

1.2 Motivação

Conforme mencionado anteriormente, nos últimos anos, muitas proposições de soluções IoHT usando padrões de movimento vem sendo apresentadas pela comunidade de Engenharia de Software e de desenvolvimento de *software* para cuidados a saúde, mas ainda existem questões em aberto a serem tratadas nesse campo de pesquisa (SYED *et al.*, 2019; YEAN *et al.*, 2018; BALAJI *et al.*, 2018; SUN *et al.*, 2020; MAJUMDER *et al.*, 2019; BRAVO *et al.*, 2018; ZEADALLY *et al.*, 2020).

Uma das questões em aberto consiste na ausência de processos bem definidos para desenvolvimento de sistemas de saúde baseados em padrões de movimentação para ambientes IoHT (SYED *et al.*, 2019). O uso de processos de desenvolvimento de *software* são fundamentais para construção de *software* de qualidade (SOMMERVILLE, 2010; PRESSMAN; MAXIM, 2016). Esses processos contém atividades que guiam todo o desenvolvimento de um *software*, desde sua definição até a implementação e evolução do produto final de *software* (PRESSMAN; MAXIM, 2016). A ausência de um processo de desenvolvimento de software dificulta a gerência das atividades de desenvolvimento e dos riscos associados ao sistema a ser desenvolvido e pode impactar negativamente na qualidade do sistema, bem como tornar difícil a execução de mudanças (SOMMERVILLE, 2010; PRESSMAN; MAXIM, 2016).

Além da ausência de processos de desenvolvimento *softwares*, ainda não há consenso sobre como relacionar os dados de movimento coletados por sensores com problemas de saúde (YEAN *et al.*, 2018; BALAJI *et al.*, 2018; SUN *et al.*, 2020; MAJUMDER *et al.*, 2019). Cada trabalho costuma apresentar soluções específicas e difíceis de reutilizar em outros estudos. Existem algumas proposições de métodos e modelos para esses tipos de sistemas na literatura (LI *et al.*, 2015; DÍAZ *et al.*, 2018; AICHA *et al.*, 2018; LAZAROU *et al.*, 2016; PEEK *et al.*, 2016), que podem ajudar a construção de novas soluções, mas ainda há espaço nesse campo de pesquisa para construção de outros artefatos que ajudem a relacionar os dados obtidos pelos sensores dos dispositivos IoHT e os diferentes problemas de saúde. Além disso, é importante construir maneiras de auxiliar na escolha de sensores e algoritmos inteligentes que permitam analisar como os dados desses sensores podem ser usados para identificar atividades, situações de risco a saúde e problemas de saúde.

Outra questão em aberto identificada na literatura é a importância de construir sistemas IoHT baseados em padrões de movimento que sejam ubíquos (BRAVO *et al.*, 2018). Uma possível solução para essa questão é a construção de sistemas IoHT autoadaptativos,

porém, como já mencionado anteriormente, não é comum encontrar proposições de sistemas autoadaptativos para soluções de IoHT e, por conseguinte, para sistemas IoHT baseados em padrões de movimento.

1.3 Hipótese, Questões de Pesquisa e Objetivo

Existem diversos processos de desenvolvimento de *software*, mas nenhum deles é útil para construção de todos os tipos de sistemas. Em geral, é comum que processos sejam adaptados para melhor construção de tipos específicos de *software* e para se adequar aos times de desenvolvimento (SOMMERVILLE, 2010).

Outra questão que deve ser considerada é que a grande maioria dos processos e modelos de processos de *software* existentes não são adequados aos sistemas autoadaptativos, uma vez que a maioria dos processos conhecidos não estão adaptados a construções de *software* que se autoadaptam (ANDERSSON *et al.*, 2013)(JUNIOR *et al.*, 2018). Porém, é possível utilizar etapas de processos de *software* conhecidos para projetar e desenvolver sistemas autoadaptativos.

Além disso, um processo de desenvolvimento *software* não provê por si só uma solução para o problema de como relacionar dados de sensores com problemas específicos de saúde. No entanto, junto aos processos de desenvolvimento de *software* é comum que existam artefatos de *software* que auxiliem a execução das diversas atividades e subprocessos existentes para cada atividade de um processo de desenvolvimento de *software*. Desse modo, um ou mais artefatos de *software* podem ser utilizados para auxiliar a resolução deste problema específico ao longo de um processo de desenvolvimento.

Considerando tudo que foi comentado, foi formulada a seguinte hipótese que será investigada nesta pesquisa:

Hipótese de Pesquisa
No desenvolvimento de aplicações IoHT baseadas em padrões de movimento, o uso de um processo de desenvolvimento de *software* e do conceito de autoadaptação é útil para facilitar o desenvolvimento dessas aplicações e prover uma maneira viável de relacionar dados de sensores e situações de saúde.

Com base na hipótese, as seguintes Questões de Pesquisa guiam essa tese:

– **QP 1 - Existem processos ou modelos de processos de *software* que podem ser adaptados para o desenvolvimento de aplicações IoHT baseadas em padrões de movimentação?** É necessário um estudo sobre processos de *software* existentes, a fim de entender

como eles podem ser usados para auxiliar o desenvolvimento de aplicações IoHT baseadas em padrões de movimento.

- **QP 2 - Quais são as estratégias que podem ser usadas para adaptar processos e modelos de processos de *software* de sistemas tradicionais para aplicações IoHT autoadaptativas e como utilizá-las?** Essa questão visa entender como os processos de *software* conhecidos na literatura podem ser utilizados para o desenvolvimento de sistemas autoadaptativos.
- **QP 3 - Que artefatos de *software* (como modelos, *frameworks*, *middlewares*, ambientes de desenvolvimento, ou estruturas de dados) podem ser utilizados para auxiliar o processo de desenvolvimento de aplicações IoHT e relacionar sensores e estados de saúde?** Essa questão tem como objetivo identificar possíveis artefatos de *software*, que possam ser utilizados para auxiliar a escolha e o uso de sensores capazes de obter dados relacionados com o estado de saúde dos usuários da aplicação.

Durante os estudos prévios feitos sobre soluções de *software* IoHT baseados em padrões de movimentação, não foi identificado nenhum processo específico ou adaptado de um processo conhecido. Além disso, não foram identificadas soluções que utilizassem sistemas autoadaptativos, ainda que fosse indicado que a falta de ubiquidade é considerada uma questão em aberto (BRAVO *et al.*, 2018). Existem na literatura artefatos de reúso para auxiliar o desenvolvimento desses *softwares* que relacionam padrões de movimento com atividades do dia a dia, quedas e alguns padrões de marcha, mas não com outros problemas de saúde (LI *et al.*, 2015; DÍAZ *et al.*, 2018; AICHA *et al.*, 2018; BALAJI *et al.*, 2018), e esses artefatos consideram um pequeno número de sensores. Considerando essas necessidades de IoHT, o objetivo geral desta tese é:

Objetivo Geral
Propor um processo de desenvolvimento de sistemas autoadaptativos e artefatos de *software* que possam ser úteis para o desenvolvimento de aplicações IoHT baseadas em padrões de movimento.

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos para os principais marcos da pesquisa dessa tese:

- Revisar a literatura e identificar modelos de processos, processos de desenvolvimento de software, ou artefatos de software, que possam ser adaptados para o desenvolvimento de sistemas IoHT autoadaptativos;

- Propor um processo de desenvolvimento de software para sistemas IoHT autoadaptativos baseados em padrões de movimento;
- Propor artefatos de software para auxiliar o desenvolvimento de sistemas IoHT autoadaptativos baseados em padrões de movimento; e
- Avaliar a viabilidade, a utilidade e a facilidade de uso do processo e dos artefatos propostos.

1.4 Metodologia de Pesquisa

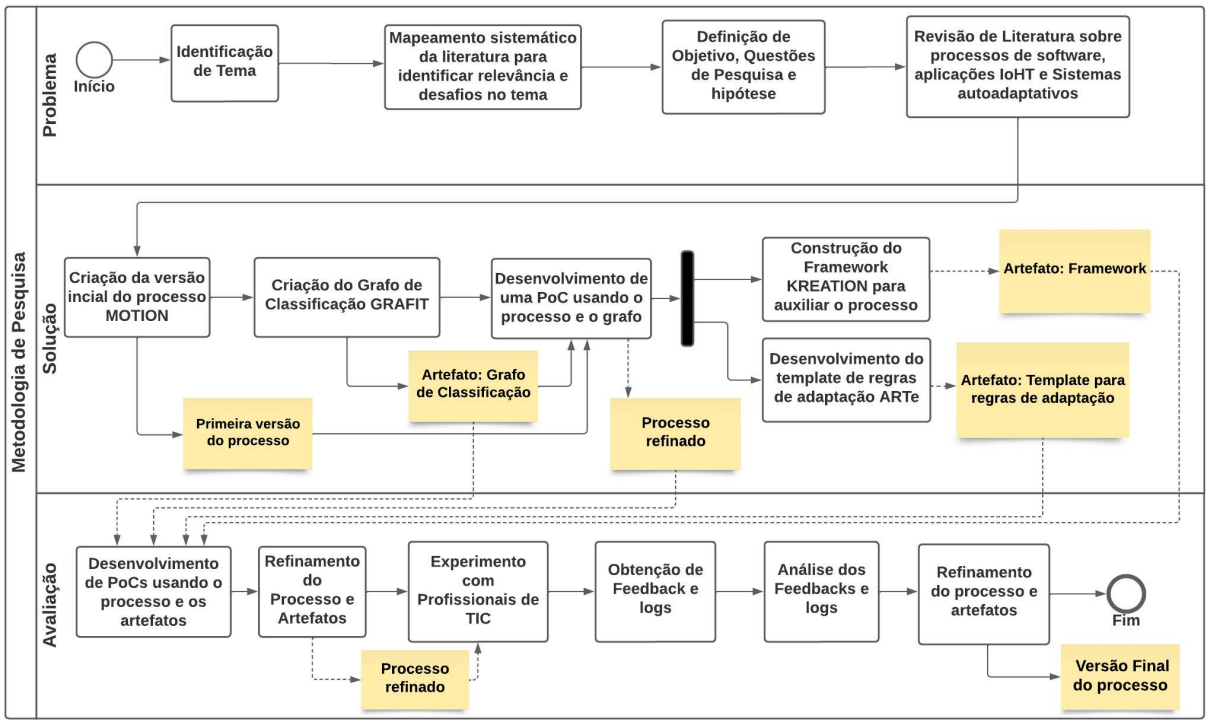
O passo a passo da metodologia de pesquisa seguida nessa tese é ilustrada na Figura 1. Esta metodologia foi inspirada no método *Technical Action Research* (TAR) (WIERINGA; MORALI, 2012) e é dividida em três fases: Problema, Solução e Avaliação.

Na fase de Problema são identificadas as questões de pesquisa e hipóteses com base em um estudo da literatura em forma de mapeamento sistemático (KITCHENHAM *et al.*, 2009) sobre soluções IoHT baseados em padrões de movimentação. Esse estudo tem como objetivo compreender como funcionam essas soluções, quais os tipos de proposta dessas soluções e quais são as questões em aberto relacionadas a esses *softwares*. Uma vez definidos o objetivo da pesquisa e qual o problema que será atacado durante a mesma, é feita uma revisão da literatura com caráter exploratório para identificar processos e/ou modelos de processo de desenvolvimento de *software* para aplicações IoHT e sistemas autoadaptativos.

Na fase de Solução é gerada a primeira versão do processo de desenvolvimento de *software* proposto, intitulado MOTION, e um modelo de grafo de classificação, intitulado GRAFIT, para relacionar sensores, *features* de movimento, algoritmos de classificação e estados de saúde. Em seguida, com base na primeira versão do processo MOTION e do GRAFIT é desenvolvida uma prova de conceito (HORTON; RADCLIFFE, 1995), do inglês *Proof of Concept* (PoC), que foi utilizada para avaliar a versão inicial do processo MOTION e do grafo de classificação. Em seguida, com base nos estudos prévios da literatura e nas lições aprendidas com o desenvolvimento da PoC, são projetados e desenvolvidos os demais artefatos de *softwares* propostos, incluindo o *template* ARTe, que auxilia a construção e utilização das regras de adaptação para aplicações autoadaptativas, e o *framework* KREATION, para auxiliar a implementação de aplicações IoHT autoadaptativas para dispositivos Android. Por fim, ainda nessa fase, o processo proposto e os artefatos são refinados.

Na última fase foi feita a avaliação do processo e dos artefatos propostos utilizando uma segunda PoC, contemplando duas aplicações desenvolvidas utilizando a versão refinada

Figura 1 – Metodologia de pesquisa



Fonte: Próprio Autor

do processo MOTION, o *template* ARTe e o *framework* KREATION. Finalmente, o processo MOTION foi avaliado através de um experimento com profissionais de Tecnologia de Informação e Comunicação (TIC). O experimento seguiu os preceitos da Engenharia de Software Experimental (WHOLIN *et al.*, 2000; WOHLIN *et al.*, 2012), e tem como objetivo avaliar o uso do processo MOTION por profissionais de TIC, com o auxílio dos artefatos de reúso propostos, através da experiência de desenvolvimento de duas aplicações IoHT autoadaptativas baseadas em padrões de movimento.

1.5 Estrutura do documento

O restante desta Tese de Doutorado está estruturado da seguinte forma:

- o **Capítulo 2** apresenta a fundamentação teórica necessária para o entendimento do processo MOTION e dos demais artefatos de *software* propost]os;
- no **Capítulo 3** são apresentados os trabalhos relacionados a Tese;
- no **Capítulo 4** é descrito o processo MOTION e os artefatos de *software* desenvolvidos;
- no **Capítulo 5** é apresentada a avaliação do processo MOTION e dos artefatos de *software* desenvolvidos; e
- o **Capítulo 6** conclui este documento, sumarizando a pesquisa desenvolvida e as propostas

de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a fundamentação teórica da pesquisa. Primeiramente são apresentados os conceitos básicos que envolvem o desenvolvimento de aplicações de *Internet of Health Tings* (IoHT) na Seção 2.1; já na Seção 2.2 são explorados os principais elementos dos sistemas de IoHT que utilizam padrões de movimentação; na Seção 2.3 são mostrados os conceitos básicos de processos de desenvolvimento de *software* e o processo de desenvolvimento de *software* baseado em reúso; na Seção 2.4 é detalhado o modelo de processo orientado a reúso de componentes; os sistemas autoadaptativos são apresentados na Seção 2.5; na Seção 2.6 são apresentados o conceito de *framework* e os métodos para construção desses artefatos de reúso; e finalmente na Seção 2.7 é feita a conclusão do capítulo.

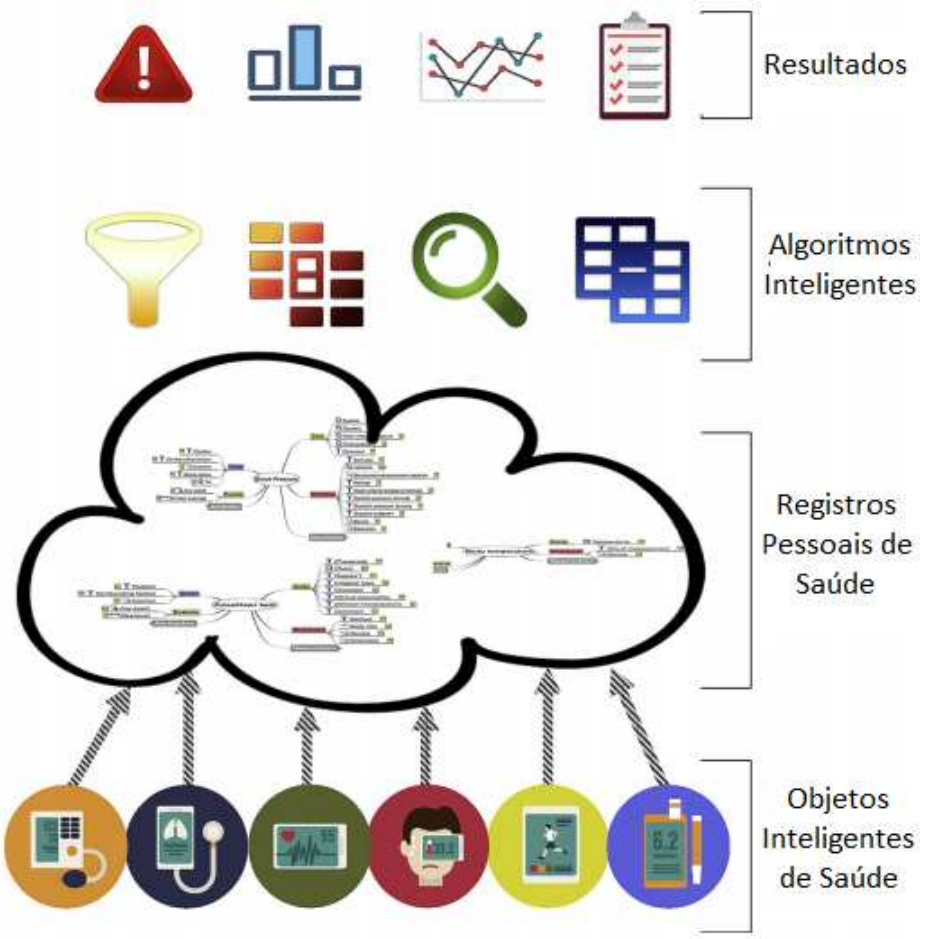
2.1 *Internet of Health Things*

A Internet das Coisas Médicas (do inglês, *Internet of Health Things* (IoHT)) engloba todo o conjunto de soluções baseadas em Internet das Coisas (do inglês, *Internet of Things* (IoT)) voltados aos cuidados a saúde, incluindo os sistemas hospitalares baseados em IoT, os sistemas de monitoramento de dados vitais, como o nível de oxigênio no sangue e o pulso sanguíneo, que utilizam dispositivos e sensores de IoT e os sistemas que usam dados de sensores diversos, como dados de movimentação, para monitorar a situação de saúde e prover melhor qualidade de vida aos pacientes (RODRIGUES *et al.*, 2018).

Segundo (COSTA *et al.*, 2018), a IoHT consiste em objetos interconectados com a capacidade de trocar e processar dados para melhorar a saúde do paciente. Esta visão centrada no paciente envolve quatro camadas distintas, como ilustrado na Figura 2.

- **Aquisição**: esta camada consiste no uso dos Objetos Inteligentes de Saúde (OIS), como dispositivos médicos e vestíveis. O objetivo de um OIS é coletar dados de sinais vitais ou de outras condições fisiológicas do paciente. Eles devem possuir recursos de comunicação com a internet ou com uma rede privada (por exemplo, Bluetooth, WiFi), ou protocolos padrões (por exemplo, HL7, DICOM e outros).
- **Armazenamento**: essa camada é encarregada de representar os dados coletados em um formato altamente escalável e interoperável. Como defendido por (GUBBI *et al.*, 2013), é comum se utilizar computação em nuvem para esse armazenamento, dadas suas características intrínsecas, incluindo elasticidade rápida, atendimento sob demanda, amplo

Figura 2 – Visão geral de IoHT com quatro camadas, a partir da parte inferior: aquisição, armazenamento, processamento e apresentação.



Fonte: adaptado de (COSTA *et al.*, 2018)

acesso à rede e *pool* de recursos para atender a demanda escalável. Mais recentemente, devido à alta latência decorrente do uso da nuvem em alguns casos, uma variação da computação em nuvem foi defendida para IoHT. Esta variação, chamada de computação em névoa, fornece serviços de computação em nuvem de forma distribuída, integrando perfeitamente dispositivos locais (às vezes chamados de computação de borda) com recursos remotos na nuvem (DASTJERDI; BUYYA, 2016; VASCONCELOS *et al.*, 2019). Outra característica importante é o registro das informações do paciente por meio de um prontuário/registro de saúde eletrônico com interoperabilidade semântica.

– **Processamento**: trata-se da camada responsável pela análise dos dados do paciente. É comum nessa camada a utilização de algoritmos inteligentes baseados em técnicas de aprendizado de máquina, de modo, a prover um processamento inteligente, que permita otimizar recursos.
– **Apresentação**: essa camada é responsável pela visualização dos resultados como uma

combinação das camadas anteriores. Essa visualização pode assumir a forma de alertas, ações sugeridas, gráficos e tabelas. As vistas epidemiológicas podem ser obtidas combinando dados não identificados de diferentes prontuários de saúde eletrônicos dentro de um contexto particular.

2.2 Sistemas IoHT baseados em padrões de movimentação

Nos últimos anos, muitos estudos tem proposto soluções para sistemas IoHT baseados em padrões de movimentação (MAGNO *et al.*, 2018; HOSSAIN; INOUE, 2019; PARK *et al.*, 2020; PEIMANKAR *et al.*, 2023). Esses sistemas coletam dados de sensores que podem ser usados para identificar padrões de movimentação relacionadas a marcha, postura, velocidade do movimento e até localização (TARAMASCO *et al.*, 2018; MOCK *et al.*, 2007; MAGNO *et al.*, 2018), a fim de monitorar pacientes e identificar problemas de saúde, como doenças crônicas, ou situações de risco, como quedas (LINHARES *et al.*, 2020; LI *et al.*, 2022; ARAÚJO, 2022), ou ainda prover um auxílio a pessoas com problemas de mobilidade (VASYLKIV *et al.*, 2019).

Os sistemas propostos por esses estudos, não costumam seguir um processo específico, mas algumas etapas são comuns a esses sistemas. As principais etapas em comum seguidas por esses sistemas consistem na aquisição de dados, seguido pelo tratamento desses dados para extração das *features* de movimentação, e pela avaliação das *features* com o objetivo de identificar os padrões de movimento (MOHAMED *et al.*, 2017; BARRY *et al.*, 2018; GIBSON *et al.*, 2016).

A aquisição é feita com base nos diversos sensores dos dispositivos IoT, que podem está em dispositivos móveis ou em *wearables*, que é o caso da maioria dos estudos, ou dispostos no ambiente. Uma vez que os dados sejam capturados pelos sensores, os mesmos costumam passar por um tratamento para que sejam extraídas as *features* que irão caracterizar os padrões de movimento. Essas *features* podem ser extraídas através de cálculos simples, ou uso técnicas mais sofisticadas como o *Dynamic Time Warping* (SANSRIMAHACHAI *et al.*, 2018), que permite analisar séries temporais. Com base nas *features* é feita uma análise que permite reconhecer os padrões de movimentação e possíveis movimentos, ou estados de saúde, como doenças crônicas (MAGNO *et al.*, 2018). Para se fazer essa análise podem ser utilizados algoritmos de aprendizagem de máquina baseados em experimentos (PARK *et al.*, 2020), ou ainda pode ser feita a verificação de *thresholds* em relação aos valores das *features* ao longo de um período de tempo (LINHARES *et al.*, 2020).

2.3 Processos de desenvolvimento de *software*

Um processo de desenvolvimento de *software* é definido como um conjunto de atividades que leva à produção de um produto de *software* (SOMMERVILLE, 2010). Segundo (PRESSMAN; MAXIM, 2016), o processo é o alicerce da engenharia de *software*. É ele que permite o desenvolvimento racional e oportuno dos sistemas de *software*.

Não existe um processo ideal para todos os tipos de *software*. Por exemplo, para alguns sistemas críticos com requisitos bem definidos, como um sistema de controle de vôo, é necessário um processo de desenvolvimento muito bem estruturado e rígido. Já para sistemas de negócio com requisitos que mudam rapidamente, um processo flexível e ágil é provavelmente mais eficaz (SOMMERVILLE, 2010).

Ainda de acordo com (SOMMERVILLE, 2010; TULIO, 2020), existem inúmeros processos de *software*, mas algumas atividades são comuns entre eles: Especificação, Projeto e Implementação, Validação de *software* e Evolução. Durante a atividade de Especificação são definidas as funcionalidades do *software* e as restrições sobre sua operação. Na atividade de Projeto e Implementação é desenvolvido o *software* que atenda as especificações. A atividade de Validação é responsável por garantir que *software* faça o que o cliente realmente deseja. Finalmente, na atividade de Evolução, é feita a manutenção e atualização do *software* de acordo com novos requisitos que naturalmente surgem.

Os processos de *software*, em geral, tem como base modelos genéricos que contém abstrações do processo que explicam diferentes abordagens para o desenvolvimento de *software* e que podem ser adaptados para criar processos mais específicos de desenvolvimento de *software* (SOMMERVILLE, 2010). Os modelos de processos de *software* genéricos mais conhecidos são: modelo cascata, modelo evolucionário ou incremental e o modelo de processo de *software* orientado a reúso (PRESSMAN; MAXIM, 2016; TULIO, 2020).

O modelo de processo em cascata separa em fases distintas do desenvolvimento as etapas de especificação, desenvolvimento, validação e evolução, e cada fase só é iniciada após o fim da fase anterior, de modo que o processo de desenvolvimento segue de modo sequencial. Os compromissos são assumidos no começo do processo, tornando difícil mudanças ao longo do mesmo. Por outro lado, a documentação produzida em cada fase e sua aderência a outros modelos de processos de engenharia de *software*, o torna adequado para sistemas com requisitos bem definidos e com pouca probabilidade de mudança ao longo do desenvolvimento (PRESSMAN; MAXIM, 2016).

O modelo evolucionário ou incremental intercala atividades de especificação, desenvolvimento e validação. De um modo geral, versões intermediárias do sistema final vão sendo produzidas, contemplando um grupo de requisitos do cliente a cada ciclo, de modo que novos incrementos contemplando outros requisitos vão sendo adicionados a cada versão, até que uma versão final do sistema contemplando todos os requisitos é produzida. Esta abordagem é mais eficaz que o modelo cascata para sistemas com muitas mudanças nos requisitos e atende melhor a necessidades imediatas do cliente e do usuário. Além disso, os usuários podem fornecer *feedback* sobre as versões intermediárias, enquanto o sistema está em produção, auxiliando um maior refinamento do *software* (SOMMERVILLE, 2010; TULIO, 2020).

Já o modelo de desenvolvimento de *software* orientado à reúso de componentes tem como foco a integração de componentes reusáveis. Ele permite uma redução da quantidade de *software* a ser desenvolvido, e por conseguinte pode prover redução de custo e risco. É importante destacar, que os componentes reutilizáveis devem atender aos requisitos do cliente e muitas vezes, uma adaptação desses componentes deve ser executada. Além disso, pode ser necessário um controle maior sobre a evolução do sistema, uma vez que novas versões dos componentes reusáveis que não estão sob controle da organização podem ser produzidas. É importante destacar que esse modelo de processo pode agregar elementos do modelo de *software* evolucionário, como a entrega de incrementos, ou do modelo cascata, como a documentação bem definida de cada etapa (SOMMERVILLE, 2010).

Considerando as vantagens da reutilização de componentes e a flexibilidade desse tipo de modelo de processo, optou-se por utilizar esse modelo como base para o processo proposto nesse trabalho, agregando elementos do ciclo de vida de sistemas autoadaptativos.

2.4 Modelo de Processo Orientado a Reúso de Componentes

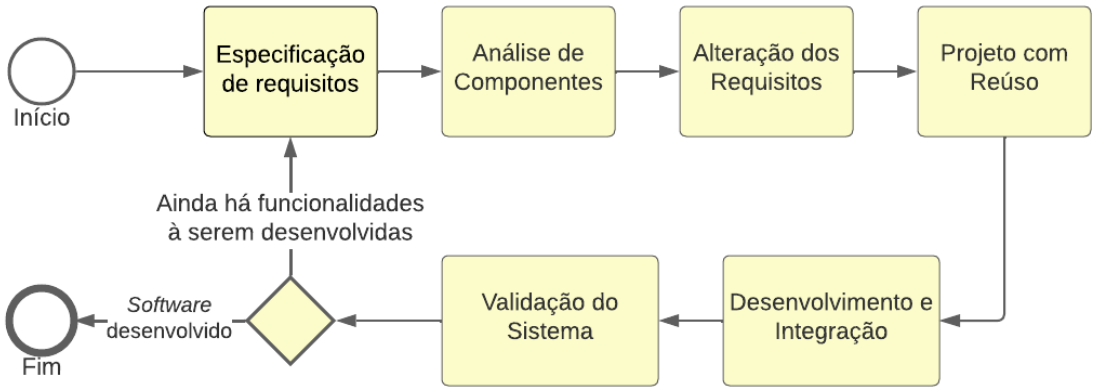
O reúso de componentes de *software* revolucionou o processo de desenvolvimento de *software*, tornando o desenvolvimento de *software* semelhante a outras indústrias de produção industrial e melhorando a eficiência do desenvolvimento de *software* (XIE *et al.*, 2020).

Com base no desenvolvimento de *software* orientado a reúso de componentes, surgiram as linhas de produtos de *software*, conjuntos de sistemas de *software* que compartilham um conjunto comum e gerenciado de recursos que satisfazem as necessidades específicas de um determinado segmento de mercado ou missão e que são desenvolvidos a partir de um conjunto comum de componentes (MARQUES *et al.*, 2019). Mais recentemente também surgiram as

linhas de produto de *softwares* dinâmicas (SANTOS *et al.*, 2017), que agregam elementos de sistemas autoadaptativos as linhas de produto de *software*.

A Figura 3 apresenta o modelo processo de desenvolvimento orientado a reúso com seis atividades intercaláveis (SOMMERVILLE, 2010). Durante a etapa de especificação é feita a análise dos requisitos e é definido quais deles serão implementados na iteração. Em seguida são analisados e escolhidos os componentes que serão utilizados. Na atividade seguinte são feitos ajustes dos componentes para atender os requisitos do software em questão e também pode ser feito ajustes nos requisitos. É então criado o projeto do sistema e feito o desenvolvimento e integração dos componentes reusáveis. Por fim são executadas as atividades de validação.

Figura 3 – Modelo de processo de desenvolvimento de *software* orientado à reúso



Fonte: Adaptado de (SOMMERVILLE, 2010)

Esse modelo de processo pode ser desenvolvido em uma ou mais iterações e é baseada no reúso sistemático em que os sistemas são integrados com componentes existentes ou sistemas *Commercial-off-the-shelf* (COTS) (MORISIO *et al.*, 2000).

Quatro estágios principais devem ser contemplados durante do desenvolvimento desses *softwares* (Figura 4): A análise e seleção de componentes reusáveis, a modificação de requisitos para contemplar o reúso de componentes, o projeto do sistema e o desenvolvimento e integração dos componentes.

O desenvolvimento de *software* baseado em componentes tem muitas vantagens, mas também tem algumas limitações de aplicação e seu desenvolvimento atual não é perfeito, uma vez que depende que hajam componentes reutilizáveis para o tipo de sistema a ser desenvolvido (XIE *et al.*, 2020). Como todo processo de *software*, o processo de desenvolvimento orientado à reúso ainda pode evoluir mais, especialmente com o estudo e criação de mais componentes de *softwares* para tipos diferentes de sistemas.

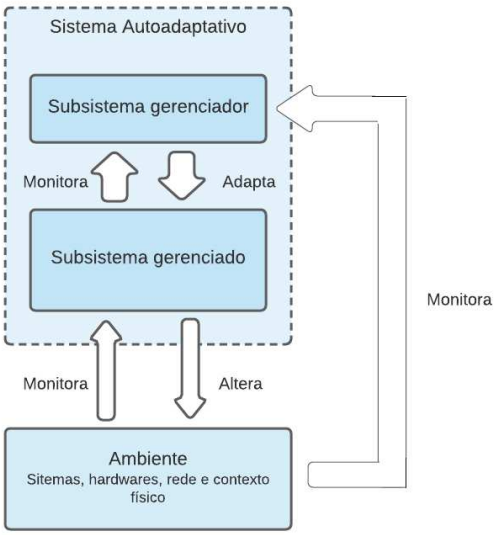
Figura 4 – Estágios do modelo de processo de desenvolvimento orientado à reúso

Fonte: (SOMMERVILLE, 2010)

2.5 Sistemas autoadaptativos

Segundo (SALEHIE; TAHVILDARI, 2009), sistemas autoadaptativos (do inglês, *Self-Adaptive System* (SAS)), são *softwares* capazes de modificar seu próprio comportamento em resposta às mudanças no seus contexto. Pelo contexto entende-se o ambiente no qual o sistema está inserido, incluindo quaisquer itens observáveis do sistema, tais como: entradas de usuário, dispositivos externos de *hardware*, sensores, entre outros. De acordo com (BRUN *et al.*, 2009), um sistema autoadaptativo é aquele que possui capacidade de modificar autonomicamente o seu próprio comportamento, em resposta a estímulos e para tal, deve haver um monitoramento do próprio sistema e do ambiente no qual ele está executando. Deste modo, tomar decisões com base nos estímulos detectados e adaptar-se.

Figura 5 – Elementos de um sistema autoadaptativo



Fonte: Adaptado de (WEYNS *et al.*, 2013) e (JUNIOR *et al.*, 2018)

De acordo com (OREIZY *et al.*, 1999), o ponto chave dos SAS, é que o ciclo de

vida do *software* nunca termina, já que o *software* precisa constantemente se autoavaliar e alterar seu comportamento de acordo com as mudanças do ambiente e exigências de usuário. Durante seu ciclo de vida, o SAS necessita do mínimo possível de intervenção humana, interagindo com o usuário do sistema apenas quando necessário ou quando é estritamente solicitado pelo usuário. Para (JUNIOR *et al.*, 2018) a auto-adaptação está ligada à criação de modelos capazes de descrever o comportamento e a estrutura do sistema, que depois se traduzem em módulos de execução que implementam os requisitos do sistema. Desse modo, a mudança do comportamento de um SAS, envolve a análise dos modelos que descrevem o estado do sistema de acordo com regras de adaptação, e ações de melhoria que são implementadas alterando módulos inadequados e ajustes de parâmetros de execução.

De um modo geral, um SAS é composto por um subsistema gerenciado, que contém a lógica da aplicação, e um subsistema gerenciador, que contém a lógica de adaptação (Figura 5) (WEYNS *et al.*, 2013).

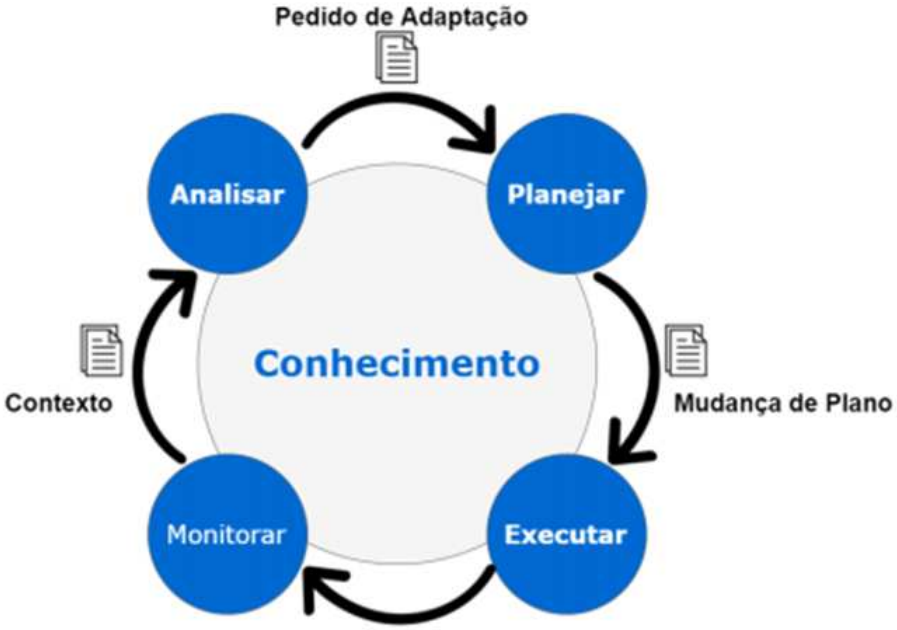
2.5.1 Loop de adaptação

Segundo (INVERARDI; TIVOLI, 2007), para que a adaptação do sistema ocorra de forma correta, é necessário responder as seguintes questões: (i) *why* - Porque é necessário adaptar? (ii) *what* - O que é preciso adaptar? (iii) *when* - Quando a adaptação deve acontecer? (iv) *who* - Quem gerencia a adaptação?. A primeira questão refere-se à causa por traz da adaptação. Geralmente é necessário adaptar devido a algum requisito que não está sendo atendido. A segunda questão refere-se a que componente do sistema deve ser alterado. A terceira está relacionada com o momento no ciclo de vida do *software* em que a adaptação deve ocorrer. Por fim, a quarta questão é relacionada a que componente gerencia as etapas necessárias à adaptação do *software*.

De maneira a garantir que a adaptação responda as questões acima a ser consideradas, os sistema autoadaptativos seguem um *loop* de controle para o processo de adaptação conhecido como MAPE-K (Figura 6) e através desse modelo é possível coordenar, manter e evoluir o sistema em tempo de execução (COMPUTING *et al.*, 2006; SALEHIE; TAHVILDARI, 2009). O *loop* MAPE-K possui quatro fases: Monitoramento (M), Análise (A), Planejamento (P) e Execução (E). No centro, circundado por quatro fases está o Conhecimento (do inglês, *Knowledgment* (K)), que representa o conhecimento necessário para auxiliar adaptação.

Durante a fase de monitoramento é feita a coleta e a correlação do dados de sensores.

Figura 6 – *Loop* de adaptação MAPE-K



Fonte: (JUNIOR *et al.*, 2018)

Essa fase auxilia na resposta sobre onde ocorreu a mudança, quando ocorreu a mudança, e o que foi alterado. A fase de análise é responsável por analisar obtidos na fase anterior e identificar onde, quando e porque deve ocorrer a mudança. A fase de planejamento determina o que deve ser alterado e como deve ser feita a alteração para alcançar o estado desejável. Por fim, durante a fase de execução são executadas as ações arquitetadas na fase de planejamento (INVERARDI; TIVOLI, 2007; SALEHIE; TAHVILDARI, 2009).

O componente de Conhecimento (K) consiste de uma coleção de dados que pode conter características do sistema, sua política, requisitos, estados, objetivos e ações. Toda vez que *loop* é executado, essas informações podem ser atualizadas (JUNIOR *et al.*, 2018).

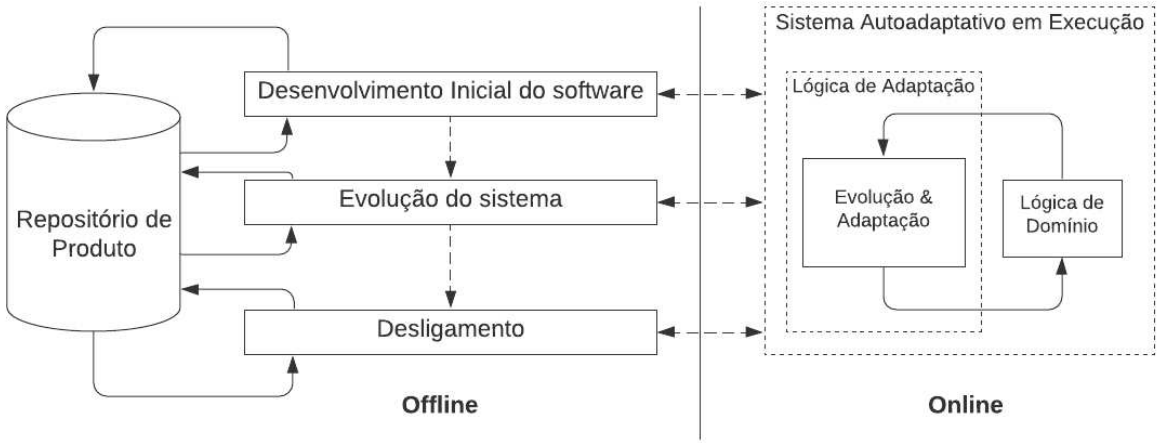
2.5.2 Ciclo de vida de um sistema autoadaptativo e processos de desenvolvimento de software

Ao escolher um processo de desenvolvimento de *software* para sistemas autoadaptativos, é preciso considerar que esses tipos de sistemas devem executar atividades do processo de *software* regulares enquanto o sistema está em execução (ANDERSSON *et al.*, 2013).

A Figura 7 ilustra como um processo de *software* e suas atividades interagem com um sistema de *software* autoadaptativo em execução (ANDERSSON *et al.*, 2013). A parte esquerda da Figura representa um modelo de ciclo de vida que abrange o desenvolvimento inicial do SAS e as atividades tradicionais de evolução e adaptação realizadas *offline*. As atividades

offline são relacionadas a artefatos, como modelos de design ou código-fonte em um repositório de produto, que não estão relacionadas diretamente ao sistema em execução e o estágio final cobre o desligamento e descomissionamento do SAS.

Figura 7 – Um modelo de ciclo de vida para sistema de *software* auto-adaptável



Fonte: Adaptado de (ANDERSSON *et al.*, 2013)

O processo da esquerda da Figura 7 é muito similar ao utilizado em um sistema de *software* tradicional. No entanto, ele pode interagir com o SAS em execução no lado direito da Figura. Essa interação ocorre por meio de atividades *online* associadas à evolução e adaptação, que constituem a lógica de adaptação do SAS. As atividades *online* evoluem e adaptam a lógica de domínio ou outra lógica de adaptação enquanto o sistema está operacional. Interações e dependências entre atividades *offline* e *online*, representadas por setas bidirecionais, são específicas para modelos de ciclo de vida visando sistemas autoadaptativos.

Além disso, a validação de sistemas de *software* é uma atividade do processo de *software* fundamental para o desenvolvimento de *softwares* de qualidade e, tal como a atividade de evolução, deve respeitar a natureza da execução contínua e adaptação desse tipo de *software* e deve acompanhar o ciclo de execução desses sistemas (SANTOS *et al.*, 2021).

2.6 *Frameworks*

Frameworks são um dos tipos de artefatos de reúso de *software* mais comuns encontrados na literatura (ARAÚJO, 2022). Eles auxiliam o desenvolvimento de sistemas, facilitando o desenvolvimento de parte ou de toda aplicação, minimizando o tempo e complexidade de desenvolvimento dos sistemas (TAHIR *et al.*, 2016). Os *frameworks* correspondem a trechos de

código semi-prontos e aptos para serem reusados (PREE, 1995), contendo funções comuns ao desenvolvimento de determinados domínios de aplicação, abstraindo do programador detalhes de implementação. Desse modo, cabe ao programador fazer ajustes no *framework* para a sua aplicação específica. Por exemplo, uma aplicação pode usar um *framework* para auxiliar o acesso ao banco de dados (SMITH, 2021), ou ainda para auxiliar a coleta de dados de dispositivos IoT (MAIA *et al.*, 2013).

Existem diversos exemplos de *frameworks* para o desenvolvimento de aplicações IoT (KHAN *et al.*, 2023; DERHAMY *et al.*, 2015) IoHT (ARAÚJO, 2022) e sistemas autoadaptivos (JUNIOR *et al.*, 2016; JUNIOR *et al.*, 2018) na literatura. Cada um desses *frameworks* tem um propósito específico e pode auxiliar o desenvolvimento de estruturas de código para diferentes tipos de aplicação.

O *framework* IoTivity (KHAN *et al.*, 2023), por exemplo, é um *framework* que tem como objetivo estabelecer a conexão dispositivo-dispositivo para aplicações IoT. Já o *framework* Ágape (ARAÚJO, 2022) tem como objetivo auxiliar o desenvolvimento de aplicações de detecção de queda e das causa da queda. Ele possui uma camada *mobile* responsável pela coleta dos dados dos sensores IoT, e uma camada em nuvem responsável pelo processamento dos dados e detecção da queda e de sua possível causa. Por sua vez, o *framework* SASeS (JUNIOR *et al.*, 2016) prover uma estrutura para o desenvolvimento de aplicações autoadaptativas baseadas em serviços, incorporando as etapas do loop de adaptação e usando o padrão *observer* para informar quando a etapa foi finalizada e a próxima pode ser executada. De forma similar, o *framework* SUCCEEd (JUNIOR *et al.*, 2018) também utiliza o padrão *observer* para comunicação entre as etapas do *loop* de adaptação, porém seu principal objetivo é auxiliar na orquestração da execução das ações e adaptações planejadas, usando um *workflow* para organizar a prioridade e ordem em que cada ação deve ser executada.

Ao longo dos anos, alguns métodos para construção de novos *frameworks* foram propostos na literatura, porém dois se destacam, o processo de desenvolvimento baseado na Análise de Domínio e o processo de desenvolvimento baseado na Experiência com Aplicações (WILSON; WILSON, 1993) (YANG *et al.*, 1998).

No processo de desenvolvimento de *frameworks* baseado na Análise de Domínio, primeiro é feita uma análise do domínio do problema para identificar e entender abstrações bem conhecidas. Essa análise extrai abstrações de aplicações existentes que são elementos comuns a todas as aplicações desse domínio e a partir delas são desenvolvidos novas aplicações de teste,

e, a partir de então, é feita a identificação das características comuns as aplicações, para que elas sejam extraídas componham o *framework*. Em seguida, as aplicações são reconstruídas utilizando o *framework* desenvolvido.

Já no processo de desenvolvimento de *frameworks* baseado na Experiência com Aplicações (YANG *et al.*, 1998), o *framework* é desenvolvido seguindo as mesmas fases comuns aos diversos processos de desenvolvimento de aplicações tradicionais, incluindo as fases de análise, projeto, implementação e teste. Na fase de análise é elaborada a especificação dos requisitos através da extração de funcionalidades comuns coletadas a partir de requisitos de aplicações similares. Na fase de projeto são identificadas as classes e os *hot-spots*, que são os elementos que variam em cada aplicação. Na fase de implementação é desenvolvido o código do *framework* de maneira repetida e incremental, assim, em cada ciclo de implementação um pequeno conjunto de requisitos é tratado e novas funcionalidades são adicionadas ao *framework*. Por fim, o *framework* é testado para saber se ele atende aos requisitos especificados e se está apto a ser reutilizado para o desenvolvimento dos tipos aplicações para o qual foi proposto.

Ambos os métodos citados anterioremente apresentam diversas vantagens e muitos autores optam por combiná-los (ARAÚJO, 2022; STANOJEVIĆ *et al.*, 2011) em um método híbrido que utiliza ideias de ambas as abordagens.

2.7 Conclusão

Este capítulo apresentou a fundamentação teórica necessária para a compreensão do trabalho proposto. Os temas da fundamentação teórica são relacionados à *Internet of Health Things*, sistemas de saúde baseados em movimento, modelos de processos de *software*, em especial, o modelo de processo de desenvolvimento de *softwares* orientado à reúso de componentes, sistemas auto-adaptativos e *frameworks*.

No capítulo seguinte são apresentados os trabalhos relacionados à esta proposta, que são resultantes de um mapeamento sistemático da literatura (KITCHENHAM *et al.*, 2009) .

3 TRABALHOS RELACIONADOS

Nesse capítulo são apresentados trabalhos identificados utilizando um mapeamento sistemático da literatura, o qual foi feito com o objetivo de pesquisar estudos de sistemas IoHT baseados em padrões de movimento para idosos. O foco desse capítulo está nos trabalhos relacionados ao estudo proposto nesse documento que apresentam métodos e artefatos de reúso tais como modelos e *frameworks*.

Na Seção 3.1 é apresentado um resumo da metodologia e do processo de mapeamento sistemático da literatura sobre sistemas de *software* IoHT baseados em padrões de movimento para idosos; a Seção 3.2 apresenta trabalhos focados em métodos para desenvolvimento de sistemas IoHT de monitoramento de saúde baseados em padrões de movimento; já na Seção 3.3 são apresentados estudos sobre Modelos para auxiliar a classificação de padrões de movimento; na Seção 3.4 são apresentados *Frameworks* para o desenvolvimento de sistemas IoHT baseados em padrões de movimento; e, finalmente, na Seção 3.5 é exposta a conclusão do capítulo.

3.1 Mapeamento Sistemático da Literatura

Esse mapeamento sistemático da literatura seguiu a metodologia proposta por (KITCHENHAM *et al.*, 2009), que descreve as etapas para um mapeamento e revisão sistemática da literatura na área de Engenharia de Software. Este método tem três atividades: Planejamento, Execução (ou condução) e Apresentação (ou documentação). Na fase de planejamento, são definidas as questões de pesquisa e a estratégia de busca, e é gerado um protocolo de mapeamento sistemático para orientar as demais fases da pesquisa. Na fase de condução, a literatura é pesquisada utilizando a estratégia de busca escolhida, os critérios para seleção dos estudos são aplicados e os dados são extraídos e sintetizados. Por fim, na fase de documentação, é feito o relatório final.

Esse mapeamento foi conduzido no primeiro trimestre do ano de 2020 e estendido em junho de 2023, tendo como principal objetivo obter um panorama dos últimos anos de pesquisas de sistemas IoHT baseado em padrões de movimento, com foco principalmente em sistemas de saúde para idosos, que é o público alvo principal desses sistemas. Além disso, buscou-se identificar tendências, desafios e questões em aberto nesse campo de pesquisa.

A estratégia de busca da literatura utilizada para esse mapeamento consistiu em uma busca na base de dados de estudos científicos SCOPUS, que é uma das principais bases de

pesquisa para a área de ciência da computação (ARCHAMBAULT *et al.*, 2009; CHADEGANI *et al.*, 2013). Essa base agrega trabalhos científicos de várias bases acadêmicas incluindo os principais *journals* e conferências de computação. A seguinte *string* de busca foi utilizado para obtenção dos estudos na base SCOPUS:

string de Busca
TITLE-ABS-KEY ((“motion” OR “movement” OR “gait” OR “mobility”) AND (“Smart Health” OR “E-health” OR “Ambient Assisted Living” OR “AAL” OR “Tele-healthcare” OR “Telemedicine” OR “Healthcare” OR “Health Promotion”) AND (“older adult*” OR “elderly”) AND (“Trend*” OR “Challenge*” OR “Open Question*”)) AND (LIMIT-TO (PUBYEAR, 2023) OR (LIMIT-TO (PUBYEAR, 2022) OR (LIMIT-TO (PUBYEAR, 2021) OR (LIMIT-TO (PUBYEAR, 2020) OR LIMIT-TO(PUBYEAR, 2019) OR LIMIT-TO(PUBYEAR, 2018) OR LIMIT-TO(PUBYEAR, 2017) OR LIMIT-TO(PUBYEAR, 2016) OR LIMIT-TO(PUBYEAR, 2015)) AND (LIMIT-TO(LANGUAGE, “English”))

Note que os termos IoT ou IoHT não estão presentes na *string*, pois no momento da busca, foi identificado que os artigos trazidos pelos mesmos já eram contemplados pelos demais termos. Também vale destacar que foram avaliadas pelo menos mais cinco *strings* antes de se chegar a essa *string* de busca final.

Com base na *string* de busca foram encontrados 366 estudos, sendo quatro deles duplicados. Para os 362 artigos restantes foram avaliados critérios de elegibilidade (exclusão e inclusão) com o objetivo de selecionar estudos primários, secundários e terciários, que apresentassem soluções IoHT utilizando padrões de movimentação com foco em idosos, publicados entre o anos de 2015 e 2023. Ao fim dessa avaliação foram selecionados 113 estudos. Todos estes estudos foram lidos por completo, porém, foi verificado que 24 deles também não cumpriram todos os critérios de elegibilidade. Logo, ao fim da fase de condução, foram extraídas e sintetizadas as informações de 89 artigos. Todas as etapas do processo de condução foram executadas por dois pesquisadores e, para garantir que os pesquisadores estavam alinhados, foi utilizada a técnica Kappa (VIEIRA *et al.*, 2010), obtendo um valor de concordância superior a 75%, um nível de concordância considerado bom para esse tipo de pesquisa.

Dentre os resultados obtidos nesse mapeamento sistemático, foram identificados os desafios de pesquisa que guiam esta pesquisa. Também foi possível averiguar que existem certas etapas comuns para aplicações IoHT baseadas em padrões de movimento, como, por exemplo, a aquisição dos dados pelos sensores, o pré-processamento desses dados, o uso de técnicas de extração de *features* e o uso de servidores em nuvem para processamento de algoritmos de aprendizado do máquina usados para relacionar as *features* com os padrões de movimento,

situações de risco a saúde e reconhecimento de atividades físicas.

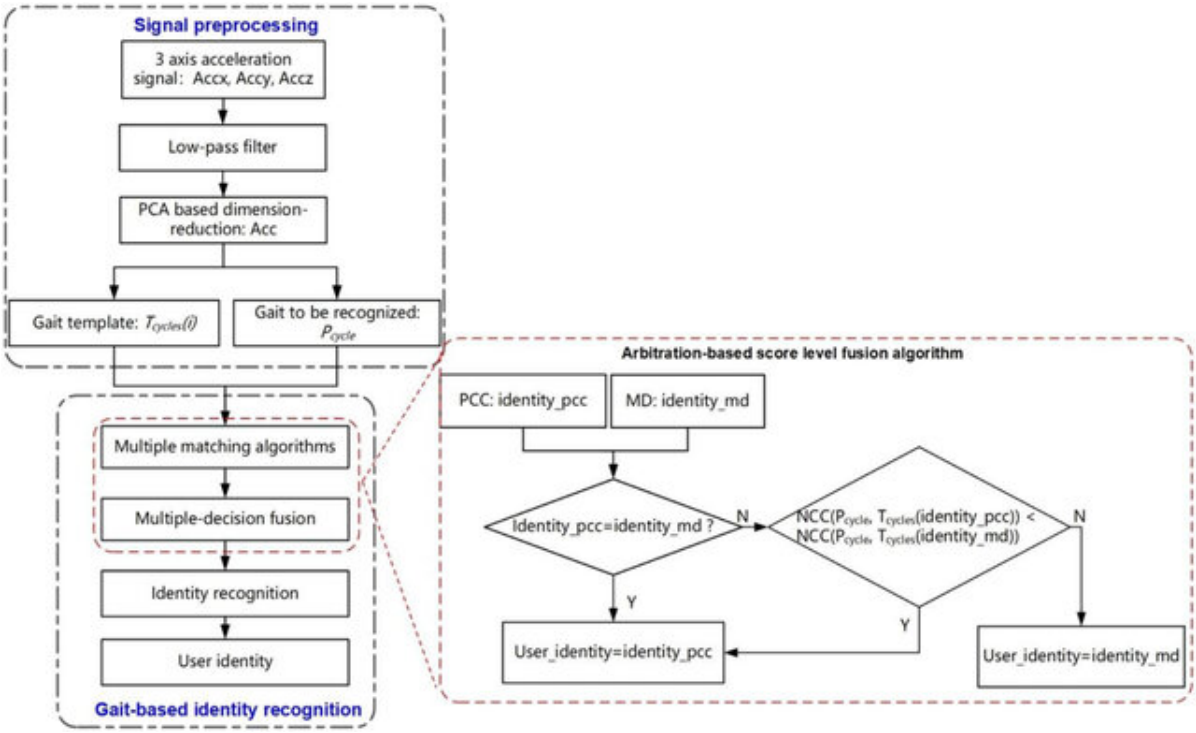
Como resultado desse mapeamento sistemático da literatura também foram identificados alguns métodos, modelos, incluindo modelos de grafos, e *frameworks* para auxiliar o desenvolvimentos de aplicações IoHT. Esses estudos são descritos nas seções seguintes desse capítulo.

3.2 Métodos

3.2.1 Gait-based identification for elderly users in wearable healthcare systems

O estudo de (SUN *et al.*, 2020) apresenta uma método para identificar um paciente, com base em informações de marcha obtidas com a análise dos dados captadas por sensores de dispositivos *wearables*. Esse estudo visa melhorar o reconhecimento de dispositivos *wearables* para idosos, através da identificação de indivíduos baseado na análise do padrão de marcha, a fim de permitir o desenvolvimento de soluções de saúde mais seguras para esse público.

Figura 8 – Método para identificação de usuário usando padrões de movimento



Fonte: (SUN *et al.*, 2020)

O método (Figura 8) utiliza dados de um acelerômetro que são pré-processados usando o algoritmo *Principal Component Analysis* (PCA) para extrair as *features* de movimentação. Em seguida, são utilizados os algoritmos para cálculo do *Pearson's Correlation Coefficient*

(PCC) e da *Manhattan Distance* (MD), que são usados para a avaliação preliminar da identidade do usuário, enquanto a decisão final de aceitar ou rejeitar a identidade do usuário é tomada por um algoritmo de correlação cruzada de normalização.

É feito um experimento para avaliação do método com uma base de dados de 744 pessoas entre 2 e 78 anos, sendo 64 delas com mais de 50 anos de idade. Os resultados experimentais mostraram que a taxa de reconhecimento por meio do método proposto foi melhorada em 26,7% em média, em comparação com uma correspondência simples baseada em PCC. Os autores propuseram como trabalho futuro a extensão do estudo para abordar o reconhecimento da marcha de crianças e pacientes com doenças musculares e nervosas, cujos padrões de marcha são instáveis.

3.2.2 Depth Maps-Based Human Segmentation and Action Recognition Using Full-Body Plus Body Color Cues Via Recognizer Engine

Esse estudo de (JALAL *et al.*, 2019) propõe um método para segmentar, rastrear e reconhecer movimentos humanas usando imagens de silhuetas de profundidade, obtidas com base em sensores que detectam variação de profundidade em imagens, como o Kinect (ZHANG, 2012).

O método funciona da seguinte maneira, um Kinect é utilizado para obter as imagens de profundidade. Depois o plano de fundo da imagem é extraído usando a técnica de Mapas de ruído de Profundidade, então é feita uma extração de *features*, onde são separadas as silhuetas e as articulações na imagem. Finalmente é feita a classificação das ações usando o algoritmo *K-means* e posteriormente usando *Hidden Markov Models* (HMM).

O método é avaliado com base em um experimento feito com um *dataset* público de imagens com cerca de 20 tipos diferentes de movimento. O método proposto foi comparado com os estudos (LI *et al.*, 2010; JALAL *et al.*, 2018; JALAL *et al.*, 2017) que propõe soluções específicas para reconhecimento de padrões de movimento e em todas as situações avaliadas, o método proposto mostrou resultados percentuais melhores para identificação dos padrões de movimento do que os demais estudos.

3.2.3 Activity Classification in Independent Living Environment with JINS MEME Eyewear

Esse trabalho foi desenvolvido por (DÍAZ *et al.*, 2018) e apresenta uma análise do uso de um óculos inteligente (*JIMS MEME*) com sensores de eletro-oculografia, giroscópio e

acelerômetro, para detecção de atividades do dia a dia (em inglês, *Activity Daily Living* (ADL)) e um método para monitoramento de movimentos e atividades oculares.

O método utiliza os dados obtidos do óculos para rastreio de movimentos da cabeça e dos olhos. Esses dados são filtrados para diminuir o ruído das medições e é avaliado o uso da técnica SMOTE (CHAWLA *et al.*, 2002) para aumentar o volume de dados. São então utilizados quatro algoritmos de reconhecimento de padrões para identificação das atividades com bases nos dados filtrados: IBk, PART, Random Forest e Sequential Minimal Optimization. Com esse método é possível identificar diferentes ADLs, tanto atividades básicas (como se vestir, tomar banho, comer e andar) quanto atividades instrumentais (como Cozinhar e assistir tv).

Foi conduzido um estudo experimental com doze idosos saudáveis, que executaram 10 diferentes atividades e foi analisado o uso do método com e sem a técnica de SMOTE. Em todas as situações, o método se saiu melhor com o uso da técnica, obtendo precisão superior a 90% na identificação das atividades.

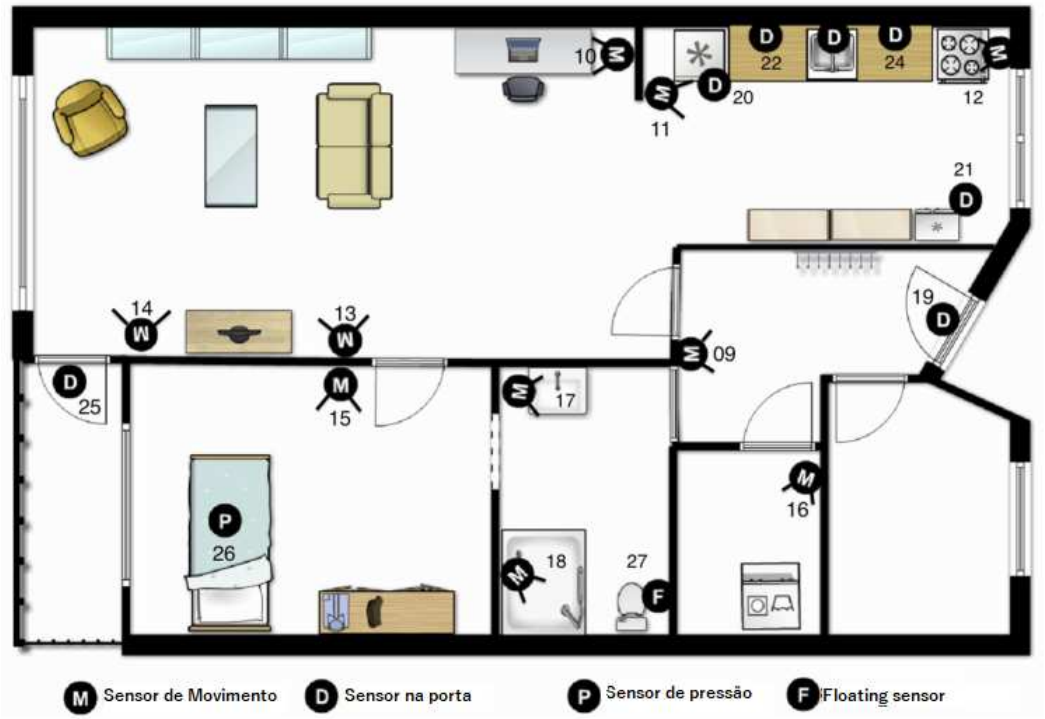
3.2.4 Continuous measuring of the indoor walking speed of older adults living alone

Em (AICHA *et al.*, 2018) é apresentado um método para calcular e estimar trajetórias seguidas por idosos. Esse tipo de verificação geralmente é feita por terapeutas para acompanhar pacientes, mas os autores argumentam que esse trabalho é muito custoso. Desse modo, o estudo propõe estimar essas informações e reduzir o custo desse acompanhamento. O método utiliza principalmente sensores infravermelhos de movimento em um ambiente *indoor*, mas também utiliza sensores em portas, um sensor de pressão na cama e um *floating sensor* no banheiro (Figura 9).

Para calcular a velocidade da marcha, são gravados os caminhos da caminhada e trajetos por uma período de tempo. Esses caminhos são registrados em um grafo de trajetos, onde os nós representam os sensores ou ambientes cobertos por eles e as arestas as distâncias entre eles (Figura 10). O grafo não só modela o ambiente, mas também a duração de cada trajetória, estima a distância e permite o cálculo da velocidade média da trajetória. Com isso, é possível monitorar o paciente em um ambiente assistido e identificar mudanças na velocidade e trajetórias comumente usadas. Essas informações podem ser usadas também para auxiliar a reabilitação com terapias ocupacionais.

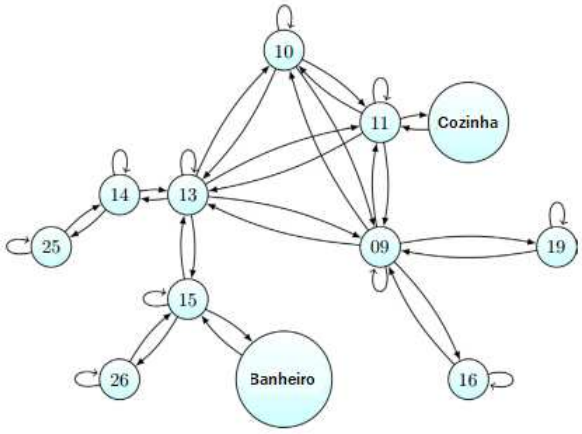
Experimentos com três voluntários são executados para avaliar diferentes aspectos da abordagem proposta. Para tal, dados dos sensores foram coletados durante 50 semanas escolhidas

Figura 9 – Mapa de um ambiente monitorado.



Fonte: Adaptado de (AICHA *et al.*, 2018)

Figura 10 – Grafo de topologia dos sensores. Os ids dos nós correspondem aos ids dos sensores representados na Figura 9. O nó da cozinha consiste em seis sensores que estão topologicamente conectados uns aos outros. Esses sensores são omitidos para manter a visão do grafo clara.



Fonte: Adaptado de (AICHA *et al.*, 2018)

aleatoriamente entre abril de 2013 e julho de 2015. Os resultados mostraram que, para cada voluntário, a estimativa contínua da velocidade da marcha com variância relativamente baixa é possível. Em comparação com os valores da velocidade de caminhada medidos por um terapeuta, a velocidade estimada pelo método foi um pouco menor, o que está dentro do esperado. Os autores deixam claro que o método não é capaz de identificar com precisão a trajetória real executada, mas faz uma estimativa com base nos dados sensoreados. Além disso, eles indicam

como possível estudo futuro, verificar se o método gera bons resultados também em ambiente doméstico com várias pessoas.

3.2.5 Automated Timed Up and Go Test for functional decline assessment of older adults

Neste estudo, (KAMPEL *et al.*, 2018) propõe dois métodos para análise de parâmetros de mobilidade durante um teste teste *Time Up & Go* (TUG) usando imagens de profundidade e análise de esqueleto captadas por um Kinect. O TUG é um teste que visa identificar problemas de mobilidade e consiste na execução repetitiva de uma série de fases predefinidas:

1. A pessoa inicia o teste sentada;
2. Em seguida a pessoa se levanta;
3. A pessoa deve então andar cerca de 3 metros em linha reta;
4. A pessoa deve fazer uma curva para retornar;
5. Em seguida, a pessoa anda em linha reta de volta até o assento;
6. Por fim, a pessoa deve se sentar novamente.

No método de análise do esqueleto, é feita leitura dos dados de cada fase do teste separadamente e é verificado o tempo de deslocamento de cada junta, para cálculo da velocidade de cada movimento (levantar, sentar, andar, virar). Já o método que utiliza a imagens de profundidade calcula a velocidade da marcha com base no centro de massa da silhueta identificada. Com base nessas análises é possível estimar se o paciente contém alguma fragilidade.

O experimento deste trabalho é executado usando imagens de bancos de dados públicos, que tiveram de ser rotuladas. Os resultados mostram que o teste TUG pode ser realizado automaticamente usando um único sensor Kinect. No entanto, os métodos apresentados podem não ser robustos o suficiente para futuros conjuntos de dados do mundo real que incluem tentativas de movimento malsucedidas ou outro comportamento atípico.

3.2.6 Comparação entre os Métodos

A Tabela 1 apresenta uma comparação entre os estudos desta Seção. Todos esses métodos utilizam dados de movimentação para auxiliar o desenvolvimento de aplicações IoHT com objetivos específicos.

Em (SUN *et al.*, 2020), os dados de movimentação coletados de acelerômetros são usados para reconhecimento de padrões de movimentos de indivíduos. Esse tipo de informação é relevante para construção de mecanismos de controle de privacidade de aplicações IoHT. Já os

Tabela 1 – Comparação entre métodos

Artigo	Sensor	Objetivo	Estrutura de Dados	Avaliação
(SUN *et al*., 2020)	Acelerômetro	Reconhecimento de indivíduos	Não menciona	Experimento
(JALAL *et al*., 2019)	Kinect	Reconhecimento de movimentos e fragilidades	Não menciona	Experimento
(DÍAZ *et al*., 2018)	Óculos Inteligentes	Reconhecimento de ADLs e movimentos oculares	Não menciona	Experimento
(AICHA *et al*., 2018)	Sensores de localização em um ambiente indoor	Reconhecimento de trajetória	Grafo	Experimento
(KAMPEL *et al*., 2018)	Kinect	Reconhecimento de movimentos e fragilidades	Não menciona	Experimento

Fonte: Próprio Autor

métodos propostos em (JALAL *et al.*, 2019) e (KAMPEL *et al.*, 2018) utilizam dados de vídeo obtidos pelo Kinect para reconhecimentos de movimentos e possíveis fragilidades. O método apresentado em (DÍAZ *et al.*, 2018) utiliza um óculos inteligente, munido de um acelerômetro, um giroscópio e sensores de eletro-oculografia para identificar ADLs e movimentos oculares e com isso verificar tanto situações anômalas de movimentos, quanto problemas oculares. Por outro lado, o método proposto em (AICHA *et al.*, 2018) utiliza sensores de localização dispostos em um ambiente *indoor* para reconhecimento de trajetórias. Entre os sensores presentes nesse trabalho estão inclusos: sensores infravermelhos, sensores em portas, um sensor de pressão posicionado em uma cama e um *floating sensor* no banheiro. O reconhecimento de trajetórias é útil para monitoramento de pessoas com problemas de mobilidade por terapeutas e para reabilitação.

As atividades de aquisição de dados e extração de *features* são comuns aos métodos propostos em (SUN *et al.*, 2020; JALAL *et al.*, 2019; AICHA *et al.*, 2018; KAMPEL *et al.*, 2018) e devem ser levadas em consideração para o desenvolvimento de soluções diversas de IoHT baseadas em padrões de movimento, desse modo devem ser considerados durante a criação do processo e dos artefatos propostos nesse trabalho.

Por fim, em (AICHA *et al.*, 2018) é utilizada uma estrutura de dados específica, um grafo, para manipulação e reconhecimento das trajetórias. Percebe-se que o uso do grafo auxiliou a correlação dos dados dos sensores com as trajetórias. De maneira similar, nesse estudo é proposto a construção de um grafo de classificação para relacionar sensores, *features* de movimento e situações de saúde.

3.3 Modelos

3.3.1 Human urban mobility classification in AAL deployments using mobile devices

O estudo de (KODYŠ *et al.*, 2017) apresenta um modelo para classificar a mobilidade baseada em sensores embarcados em *smartphones*. A solução proposta distingue a mobilidade ativa, onde o usuário quer realizar o movimento (por exemplo, caminhar, andar de bicicleta, pegar um ônibus) e a mobilidade passiva, onde o usuário usa um veículo para se movimentar (por exemplo, carro, ou táxi). O modelo também classifica a fragilidade de acordo como a mobilidade e tendo como base a classificação de um instrumento da Organização Mundial de Saúde (ORGANIZATION, 2007).

Um *dataset* com 50.000 medidas de 4 participantes e 6 tipos de movimento foi utilizado para treinamento e experimentação do modelo. Entre os movimentos estão três ativos, dois passivos e a ausência de movimento.

O modelo de aprendizado de máquina supervisionado proposto busca determinar a função definida que melhor estima o tipo de movimento. Foram avaliados modelos supervisionados que permitem rastrear uma evolução temporal. Para classificação dos dados foram usados quatro algoritmos diferentes de aprendizado de máquina: *Gaussian Naive Bayes* (GNB), *Discriminant Analysis* (DA), *Support Vector Machines* (SVM) e *K-nearest neighbors* (KNN).

Por fim, o modelo foi avaliado junto a um *framework* dos mesmo autores (TIBERGHIEN *et al.*, 2012) e os resultados demonstraram que foi possível classificar todos os tipos de movimentos como ativos ou passivos e auxiliar a identificação de fragilidades. Também foi identificado que os melhores resultados da modelagem foram obtidos com uso dos algoritmos KNN e SVM para os dados coletados.

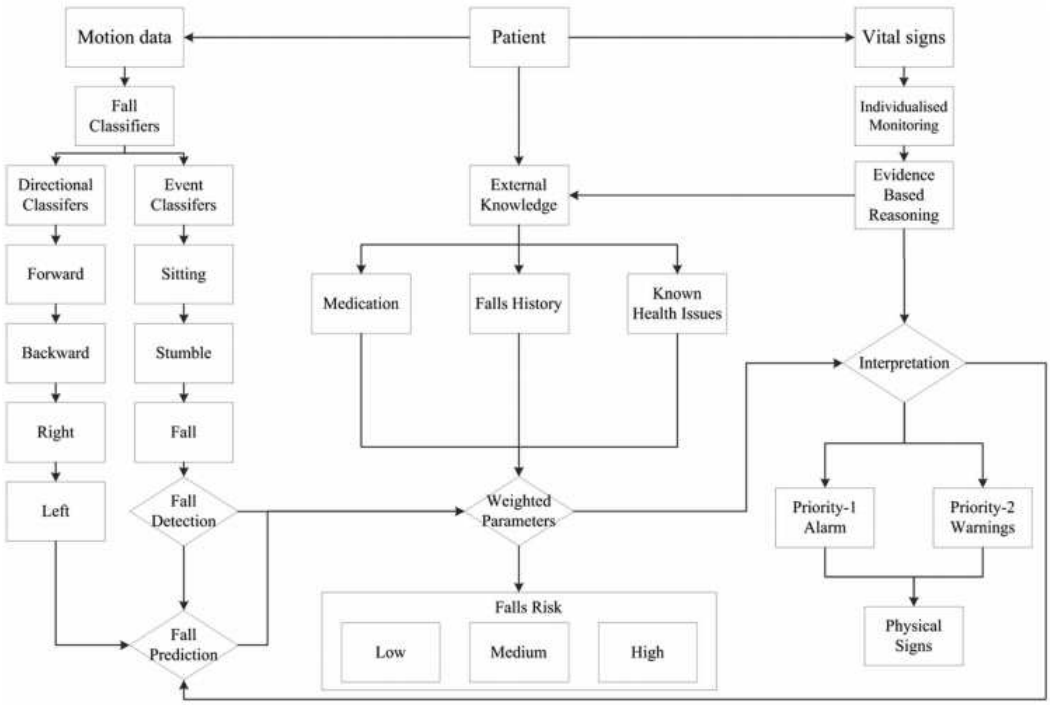
3.3.2 Falls risk assessment for hospitalised older adults: a combination of motion data and vital signs

Neste estudo, (BAIG *et al.*, 2016) propuseram um modelo para avaliação do risco de quedas em ambientes hospitalares, com o objetivo de minimizar o custo financeiro e pessoal com este tipo de injúria. A pesquisa também teve como objetivo minimizar alarmes falsos que são um incômodo para pacientes e cuidadores e podem comprometer a eficácia do atendimento.

O modelo proposto (Figura 11) classifica o risco de queda com base em padrões de movimento e sinais vitais. Os tipos de movimento, como por exemplo, caminhar, sentar, ou

uma queda, são identificados com base em padrões de dados coletados de um dispositivo com acelerômetro disposto no braço dos pacientes. A orientação do movimento também é levada em consideração, pois o tipo do movimento executado durante a queda pode indicar um risco maior ou menor a saúde. Além disso, a orientação impacta na dificuldade de detectar a queda. Já os dados vitais coletados consistem basicamente em dados de pressão sanguínea, que podem variar bruscamente de acordo com o movimento. O modelo também considera o histórico do paciente e as medicações que o mesmo utiliza, para classificar o risco da queda.

Figura 11 – Arquitetura do modelo proposto



Fonte: (BAIG *et al.*, 2016)

Foram feitos experimentos com quatro idosos saudáveis, performando atividades do dia a dia e quedas intencionais. No total foram simuladas 80 quedas e 40 atividades do dia a dia. Com base no algoritmo de detecção e classificação de quedas foram identificadas 100% das quedas com orientação frontal, 90% com orientação para trás, 85% com orientação para esquerda e 85% com orientação para direita. Foram então feitos experimentos usando o modelo proposto e o mesmo classificador de quedas usado no experimento anterior, mas com dados de 20 idosos. Como conclusão do experimento foi possível classificar quedas nos três níveis de risco do modelo.

3.3.3 DCA-IoMT: Knowledge-Graph-Embedding-Enhanced Deep Collaborative Alert Recommendation Against COVID-19

Em (KHAN *et al.*, 2022) é proposto uma abordagem de *Deep Collaborative Alert Recommendation* (DCA) usando dados de localização para lidar com a pandemia de COVID-19. O DCA coleta dados online atuais sobre o COVID-19, os purifica e os transforma em um Grafo de Conhecimento, encapsula os recursos e tendências do contexto no espaço de incorporação e os codifica para os fatores ocultos independentes usando uma rede neural de grafos.

Foram executados experimentos usando duas bases de dados do mundo real e a abordagem foi avaliada em comparação com nove outros métodos de construção de gráfico de conhecimento com base nas métricas de ganho acumulado descontado normalizado (*Normalized Discounted Cumulative Gain* (NDCG)), erro quadrático médio (*Root-Mean-Square Error* (RMSE)) e erro absoluto médio (*Mean Absolute Error* (MAE)). De acordo com os autores, os resultados mostram que a abordagem proposta superou os métodos de linha de base com melhorias finas no fornecimento de recomendações.

3.3.4 Comparação entre os Modelos

A Tabela 2 apresenta a comparação entre os modelos apresentados. Os modelos propostos em (KODYŠ *et al.*, 2017) e em (BAIG *et al.*, 2016) visam auxiliar a classificação de atividades baseadas em dado de movimentação. Porém o modelo proposto em (KODYŠ *et al.*, 2017) visa avaliar o movimento quando a ao movimento voluntário, ou a passividade (movimento involuntário), além disso, ele busca identificar situações de fragilidade com base nos movimentos classificados. Por outro lado, o modelo de (BAIG *et al.*, 2016) visa classificar e detectar o risco de quedas e para isso utiliza outros dados vitais, além dos dados de movimento que são coletados por sensores de acelerômetro. Já o modelo proposto em (KHAN *et al.*, 2022) tem como objetivo o uso de dados de localização de alertas para identificação de casos relacionados a pandemia de COVID-19.

Apesar de terem finalidades diferentes, é possível identificar em comum entre os modelos a necessidade de uma etapa de pre-processamento dos dados e extração de *features*. Esse tipo de informação reforça mais uma vez o quão importante é essa etapa para manipulação de dados pelas aplicações IoHT.

Nessa tese não é proposto diretamente um modelo de dados, porém é proposto um

Tabela 2 – Comparação entre modelos

Artigo	Sensores	Objetivo	Estrutura de dados	Local do processamento	Avaliação
(KODYS et al., 2017)	Sensores embarcados em smartphone	Modelagem de dados para reconhecimento de tipos de movimentos e fragilidades	Não define	Na nuvem ou no dispositivo	Experimento
(BAIG et al., 2016)	Wearable com acelerêmetro e medidor de pressão sanguínea	Modelagem de dados para identificar o risco da ocorrência de quedas	Não define	Na nuvem ou no dispositivo	Experimento
(Khan et al., 2022)	Sensores usados para localização (GPS, Wifi e outros)	Uso de dados de localização de alertas para identificação de casos relacionados a pandemia de COVID-19	Grafo	Na nuvem	Experimento

Fonte: Próprio Autor

modelo em forma de grafo de classificação, similar ao apresentado em (KHAN *et al.*, 2022), tendo como objetivo relacionar dados de sensores, *features*, e atividades ou estados de saúde. Desse modo, o estudo dos modelos apresentados nessa Seção contribui para melhoria desse artefato, através do estudo das etapas usadas para construção desses modelos. Além disso, o modelo de grafo proposto busca prover uma forma de modelar o uso de diferentes dispositivos e relacioná-los aos diferentes estados de saúde possíveis, tentando mitigar assim a questão da interoperabilidades dos sensores, diferentemente do que se propõe os modelos apresentados nos trabalhos relacionados.

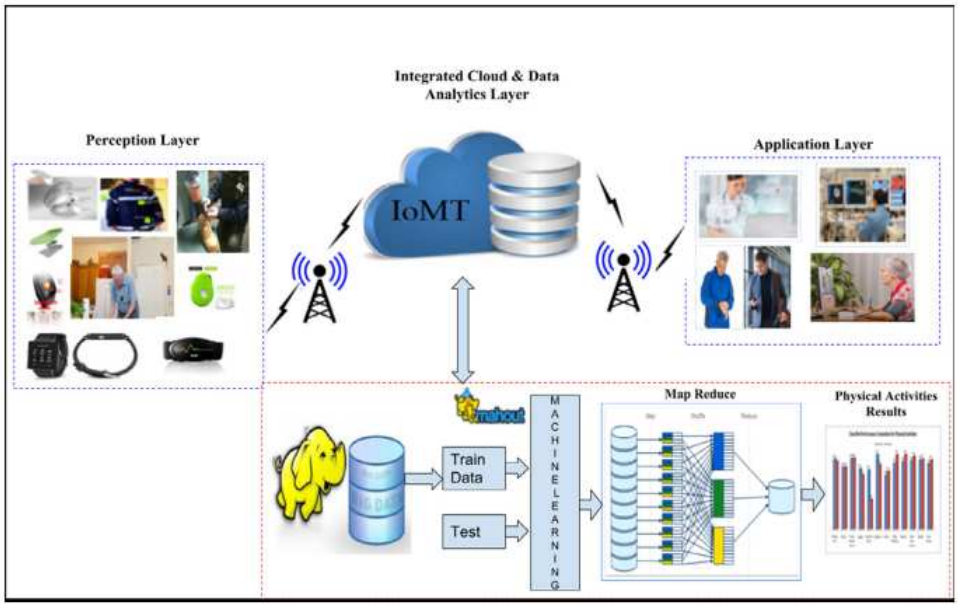
3.4 *Frameworks*

3.4.1 Smart healthcare framework for ambient assisted living using IoMT and big data analytics techniques

O *framework* proposto nesse trabalho (SYED *et al.*, 2019) (Figura 12) tem como objetivo auxiliar o desenvolvimento de aplicações de IoHT para *Ambient Assisted Living* (AAL), ambientes IoT voltados ao monitoramento de saúde e melhoria de qualidade de vida, com foco principal em idosos (BELBACHIR *et al.*, 2010; GHADI *et al.*, 2022).

O *framework* proposto permite o uso de diversos sensores embarcados em dispositivos *wearables*, que são usados para captar dados dos pacientes (Camada de Percepção). Esses dados são armazenados na nuvem usando técnicas de processamento de *big data*, especialmente hadoop (Camada de Nuvem). As *features* extraídas desses dados são analisadas para identificação da atividade usando o algoritmo de naive bayes (Camada de Análise de Dados). Finalmente essas

Figura 12 – Framework Inteligente de Saúde para AAL



Fonte: (SYED *et al.*, 2019)

features são avaliadas para reconhecimento de atividades diárias, riscos a saúde, ou possíveis doenças e os resultados são apresentados aos usuários (Camada de Aplicação).

Foram realizados experimentos com uma aplicação simples desenvolvida com o *framework* e usando 33629 registros e 12 descritores extraídos de uma base de dados de saúde obtidos de aplicativos mobile. Os resultados mostraram que a aplicação desenvolvida com o *framework* teve precisão 97,1% ao classificar as atividades físicas descritas na base de dados, o que mostra que o *framework* foi capaz de auxiliar a construção de uma aplicação IoHT para AALs com alta precisão.

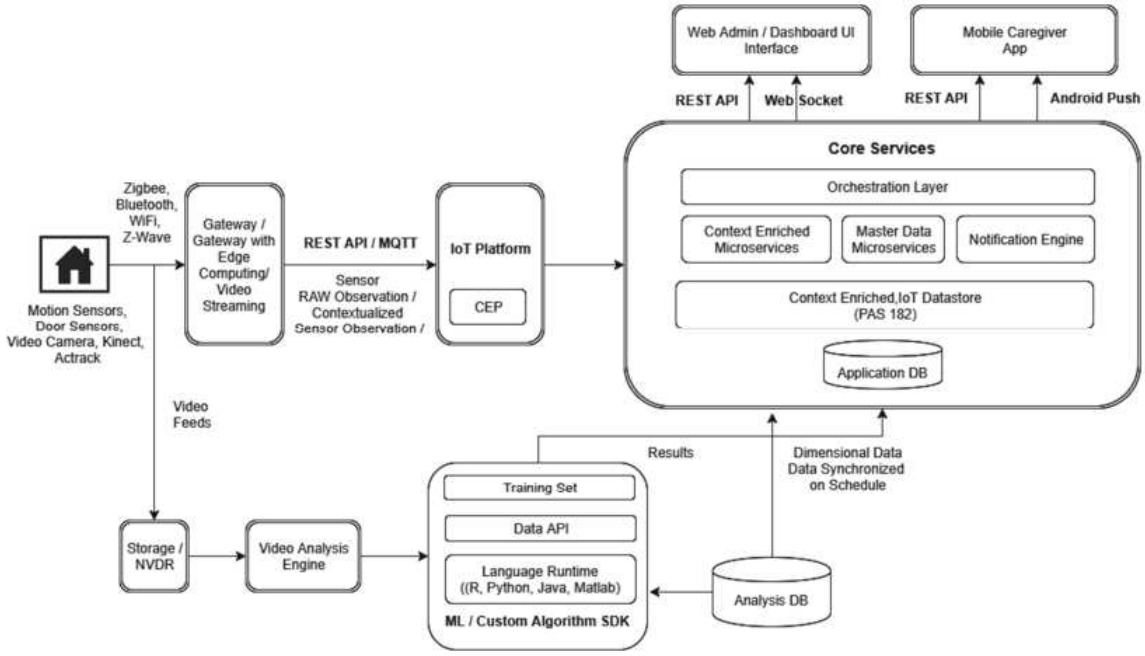
Esse estudo também apresenta desafios para construção de soluções para AALs, como a dificuldade de aceitação dos usuários idosos, a dificuldade de prover privacidade e confidencialidade dos dados, necessidade de garantir a qualidade de dados de saúde, e o isolamento que pode ser induzido pelo uso desses ambientes.

3.4.2 The Wellness Care Framework

Neste estudo, (BALAJI *et al.*, 2018) propõe o *Wellness Care Framework* para desenvolvimento de aplicações para o bem estar de idosos que suporte o uso de múltiplos sensores com bom custo benefício, não-intrusivo e com alta precisão. O objetivo dos autores foi prover um *framework* extensível e genérico que permite integrar tipos diferentes de sensores e algoritmos. Esses algoritmos podem ser executados na nuvem, na borda ou nos dispositivos.

O *framework* proposto segue a arquitetura ilustrada na Figura 13. Esse *framework* prover uma mecanismo de abstração de dados provenientes de múltiplos dispositivos, que é capaz enriquecer os dados com contexto e informações de domínio, e armazená-los em uma estrutura de dados padronizada.

Figura 13 – The Wellness Care Framework



Fonte: (BALAJI *et al.*, 2018)

O *framework* ainda oferece um *gateway* que utiliza protocolos eficientes de baixo alcance e potência para captar dados de sensores e encaminhar para uma camada de IoT que facilita a interação entre os sensores, o *gateway* e os serviços principais. A plataforma IoT também tem um componente embutido que monitora os dados e verifica um conjunto de regras baseadas em *thresholds*. Quando um *threshold* é atingido, um evento correspondente é disparado.

Os dados da plataforma IoT são então enviados para o núcleo de serviços que enriquece e armazena os dados e cria uma modelo de dados interoperável. O *hub* de dados é construído em uma arquitetura de microsserviços e oferece microsserviços de dados, microsserviços de dados mestre e microsserviços de notificação. Os microsserviços proveem APIs REST para os serviços *upstream* consumirem e renderizarem as informações em interfaces Web ou em aplicativos móveis. O *framework* ainda inclui uma camada para implantação de algoritmos personalizados, que quando executados, seus resultados podem ser disponibilizados para o restante das camadas do *framework*.

Para os dados de vídeo não é utilizada plataforma IoT, mas há um mecanismo a parte

de armazenamento. Esses dados são pré-processados em um motor de análise de dados de vídeo e depois que eles são tratados, podem ser classificados usando algoritmos de aprendizagem de máquina, ou ser utilizados para o treinamento prévio desses algoritmos, antes de serem enviados para o núcleo de serviços e estarem disponíveis para as aplicações. Por fim, vale destacar que o *framework* possui suporte as linguagens de programação R, Python, MATLAB e Java.

Três aplicações para auxiliar o monitoramento, reabilitação e a melhoria da qualidade de vidas de idosos são apresentadas no trabalho. Duas dessas aplicações utilizam o Kinect para obtenção dos dados e terceira utiliza múltiplos sensores embutidos em *smartwatches*. Há ainda uma discussão sobre o uso de vídeos para aplicações de saúde que visam a melhoria da qualidade de vida de idosos.

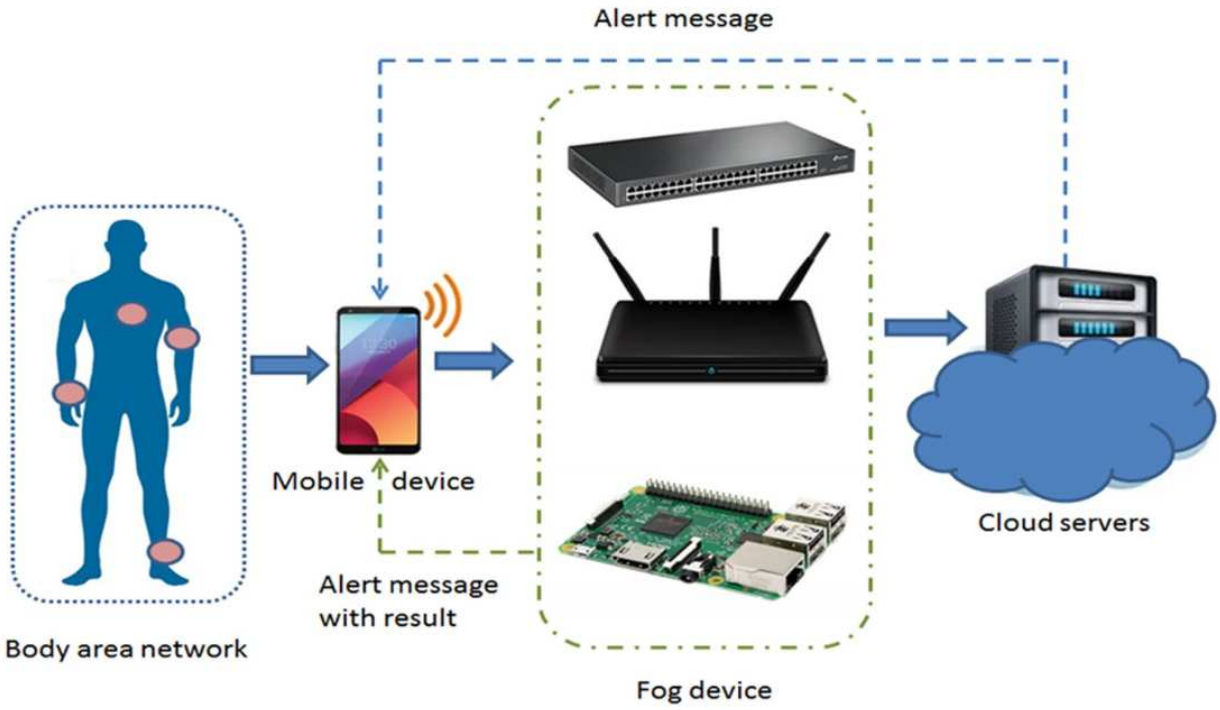
3.4.3 Internet of Health Things (IoHT) for personalized health care using integrated edge-fog-cloud network

Em (MUKHERJEE *et al.*, 2021), é proposto um *framework* para aplicações móveis IoHT que utiliza uma rede colaborativa *edge-fog-cloud*, como mostrado na Figura 14. Um grafo multicamadas é usado para modelar padrões de movimento com base na trajetória seguida pelo usuário do aplicativo. Uma rede adversária generativa também é usada para prever a sequência de localizações de acordo com o modelo do grafo.

Os dados de movimento são coletados apenas pelo sensor GPS do dispositivo, e o grafo modela as estradas ou caminhos (arestas) e suas interseções (vértices). Cada aresta é direcionada de acordo com a direção do tráfego nos caminhos. Além dos movimentos, o *framework* ajuda a coletar e identificar pressão arterial, temperatura corporal, pulsação e SpO2 para identificar situações de risco.

O *framework* proposto foi instanciado usando uma máquina no serviço de provisionamento de nuvem da Amazon EC2, e um experimento foi executado com alunos voluntários que utilizaram dispositivos móveis criados com um Raspberry PI. A previsão de movimento do *framework* foi comparada com quatro modelos inteligentes: Redes Bayesianas, Subsequência Comum Mais Longa (*Longest Common SubSequence* (LCSS)), Preditor de Markov e uma Rede Neural Convolucional. Como resultado, os autores obtiveram tempo de resposta do *framework* para identificar situações de risco menor que a previsão realizada com todos os outros modelos inteligentes avaliados.

Figura 14 – Edge-fog-cloud based IoHT framework



Fonte: (MUKHERJEE *et al.*, 2021)

3.4.4 Comparação entre os Frameworks

A Tabela 3 apresenta uma comparação entre os *frameworks* apresentados. Os *frameworks* de (SYED *et al.*, 2019), (BALAJI *et al.*, 2018) e (MUKHERJEE *et al.*, 2021) apresentados nessa Seção permitem o uso de sensores IoT diversos para o desenvolvimento de aplicações IoHT baseadas em padrões de movimento. No entanto, os *frameworks* de (SYED *et al.*, 2019) e (MUKHERJEE *et al.*, 2021) suportam o uso apenas de dados embarcados em dispositivos *wearables*, enquanto que o *framework* proposto em (BALAJI *et al.*, 2018) é mais abrangente, permitindo a manipulação de sensores *wearables* e outros tipos de sensores dispostos no ambiente, incluindo sensores de vídeo.

Tabela 3 – Comparação entre *frameworks*

Artigo	Tipos de Dispositivos IoT	Armazenamento e Tratamento de Dados	Processamento de Algoritmos Inteligentes	Mecanismos de visualização
(SYED et al., 2019)	*Wearables*	Armazenamento e Manipulação de *big data*	Nuvem	Web e *Mobile*
(BALAJI et al., 2018)	*Wearables* , Vídeo e Outros	Armazenamento e Manipulação de *big data* e Enriquecimento de Dados	Nuvem, Névoa e Borda	Web e *Mobile*
(MUKHERFEE et al., 2021)	*Wearables*	Enriquecimento e Modelagem de Dados usando Grafo	Nuvem, Névoa e Borda	Web e *Mobile*

Fonte: Próprio Autor

O *framework* de (SYED *et al.*, 2019) tem um grande foco no armazenamento e disponibilização de dados, provendo um mecanismo robusto de manipulação de *big data*. Já no *Wellness Care Framework* (BALAJI *et al.*, 2018) também são propostos mecanismos robustos para armazenamento de dados, mas o foco mais está no enriquecimento e contextualização desses dados, com objetivo garantir um melhor qualidade dos dados obtidos pelos sensores. Por outro lado, em (MUKHERJEE *et al.*, 2021) o foco não está no armazenamento de um grande volume de dados, mas na modelagem dos dados como uma estrutura da grafo, afim de facilitar a manipulação desses dados e auxiliar na inferência do movimento.

Também é possível perceber uma preocupação maior no uso de algoritmos de aprendizado de máquina para análise de features e de padrões de movimento em (BALAJI *et al.*, 2018) e (MUKHERJEE *et al.*, 2021), mas o framework proposto por (SYED *et al.*, 2019) também possui uma camadas de análise que permite a implementação de algoritmos de aprendizado de máquina diversos, embora haja menos suporte nativo para tais. Além disso, enquanto que no *framework* de (SYED *et al.*, 2019) os dados são processados na nuvem, no *Wellness Care Framework* de (BALAJI *et al.*, 2018) e no *framework* proposto em (MUKHERJEE *et al.*, 2021) há uma flexibilidade maior, uma vez que os dados podem ser processados na nuvem, na borda ou no dispositivo. Finalmente, pode-se destacar que ambos os *frameworks* possuem mecanismos de disponibilização de serviços para visualização dos dados em aplicações web e aplicações móveis.

Tal como nos estudos de (SYED *et al.*, 2019), (BALAJI *et al.*, 2018) e (MUKHERJEE *et al.*, 2021), um das proposições deste trabalho, é um *framework* para aplicações de IoHT para dispositivos móveis, no entanto, o artefato proposto nesse trabalho tem foco aplicações de IoHT autoadaptativas, que não são diretamente contempladas pelos *frameworks* apresentados nos estudos supracitados, um vez que este não consideram o *loop* de adaptação dessas aplicações em seus trabalhos. Porém, a análise desses estudos permitiu identificar uma série elementos que são importantes para *frameworks* de aplicações IoHT para dispositivos móveis, que foram incorporadas no *framework* proposto nesse trabalho, como por exemplo, a necessidade de haver um mecanismos para preprocessamento dos dados dos sensores e para extração de *features* e o processamento de algoritmos inteligentes que utilizam os dados coletados pelos dispositivos IoT.

3.5 Conclusão

Neste capítulo foram apresentados os trabalhos relacionados ao estudo proposto nesse documento, identificados como parte dos resultados de um mapeamento sistemático da literatura sobre aplicações IoHT que utilizam padrões de movimento para monitorar a saúde e melhorar a qualidade de vida de idosos. Um resumo da metodologia utilizada para a execução do mapeamento sistemático é apresentada na primeira Seção do capítulo. Nas demais seções são apresentados os trabalhos relacionados que incluem métodos, modelos e *frameworks* para sistemas IoHT baseados em padrões de movimento.

A seguir, o próximo capítulo apresenta e detalha o processo e os artefatos de reúso propostos nesse trabalho.

4 PROCESSO MOTION E ARTEFATOS DE REÚSO DE SOFTWARE

Neste capítulo é descrito o processo de desenvolvimento e os artefatos de reúso de *software* propostos nesta tese.

Na Seção 4.1 é detalhado o processo MOTION, um processo de desenvolvimento de desenvolvimento de *software* que auxilia o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Na Seção 4.2 são apresentados os artefatos a serem desenvolvidos para auxiliar o desenvolvimento de *software* utilizando o processo MOTION, incluindo: o GRAFIT, que consiste em um grafo de classificação que permite relacionar sensores e estados de saúde, o *template* ARTe, para construção de regras de adaptação, e o *framework* KREATION, para auxiliar a implementação de aplicações IoHT autoadaptativas para dispositivos Android. Por fim, na Seção 4.3 é feita a conclusão do capítulo.

4.1 Sobre o MOTION

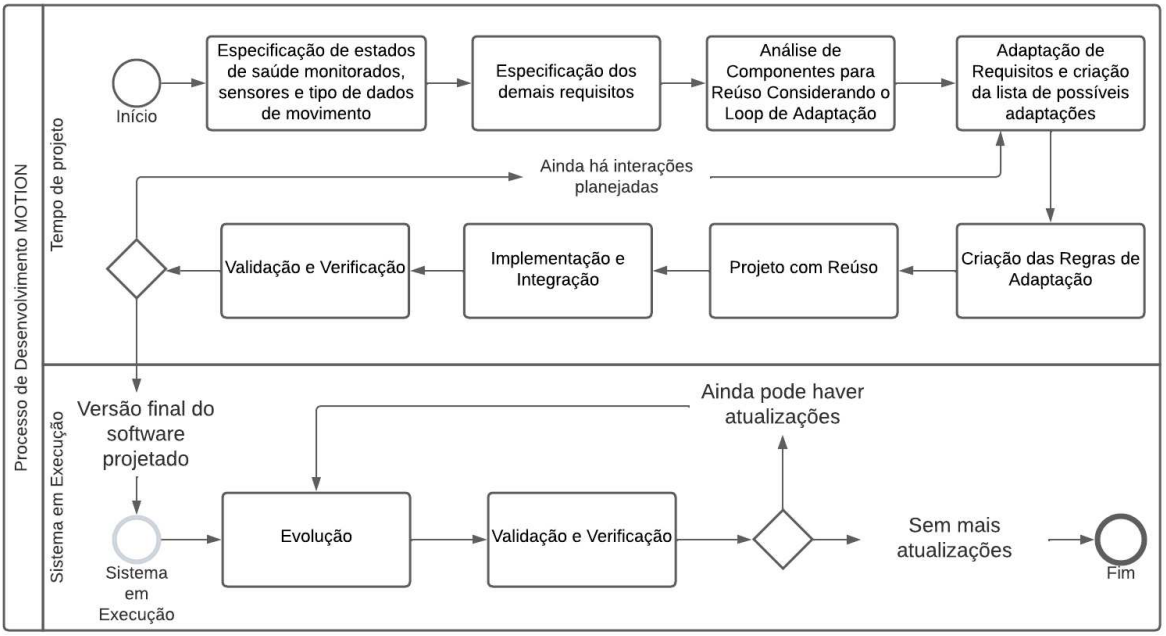
O Processo de Desenvolvimento para Aplicações IoHT Autoadaptativas Baseadas em Padrões de Movimento (MOTION), ilustrado na Figura 15, adapta o modelo de desenvolvimento de *software* orientado à reúso de componentes (XIE *et al.*, 2020; SOMMERVILLE, 2010) e adiciona atividades a esse processo, a fim de contemplar características dos sistemas autoadaptativos e dos sistemas IoHT baseados em padrões de movimento.

A motivação para o uso do modelo de processo orientado a reúso de componentes como base para o MOTION vem tanto da facilidade de agregar novos componentes reusáveis ao processo, incluindo os artefatos de *software* propostos nesse trabalho, quanto de sua flexibilidade que permite uma fácil adaptação para adicionar atividades voltadas para sistemas diversos, como os sistemas IoHT baseado em padrões de movimento. Além disso, o processo também pode ser adaptado para o desenvolvimento de sistemas autoadaptativos, com adição de algumas especificações, como é o caso dos sistemas de linhas de produtos de *software* dinâmicas (SANTOS *et al.*, 2017) que expandem o conceito de linhas de produto de *software*, que por sua vez são baseadas no modelo de processo orientado à reúso (MARQUES *et al.*, 2019).

Vale destacar que o processo proposto foi pensado inicialmente para trabalhar com dispositivos IoT móveis, especialmente *smartphones* e *smartwatches*, devido ao custo benefício desses dispositivos.

O processo MOTION é composto por dez atividades que são divididas em dois

Figura 15 – Processo MOTION



Fonte: Próprio autor

momentos: o tempo de projeto (*Design Time*) e o sistema em execução (*Runtime*).

Durante o tempo de projeto são selecionados os estados de saúde a serem monitorados, escolhidos os sensores que serão usados e levantados os requisitos. Depois, os componentes de reutilização são selecionados e as regras de adaptação são geradas. Em seguida, é feito o projeto do sistema autoadaptativo, a implementação do código e a verificação e validação.

Durante a execução do sistema, as atividades de evolução e validação ocorrem paralelamente à execução, sem a necessidade de o sistema parar de executar quando uma nova evolução ou adaptação for necessária.

As atividades em tempo de projeto podem ser executadas de maneira sequencial, como em um modelo de processo em cascata, ou de maneira intercalada, iterativa e incremental, como nos modelos de processo evolucionários. Já as atividades que ocorrem durante a execução estão relacionadas a adição de novas adaptações e a validação do sistema toda vez que uma adaptação ocorrer ou que uma nova adaptação for adicionada, de modo que elas executam de maneira cíclica junto a execução do sistema autoadaptativo.

Para guiar o uso do processo MOTION e facilitar a identificação de quais possíveis artefatos podem ser utilizados como entrada e saída de cada atividade do processo, foi gerado um manual de uso e execução do processo que pode ser visto com detalhes no Apêndice A.

4.1.1 Atividades do Processo MOTION

Nessa subseção são descritas cada uma das atividades do processo, assim como é indicado como as atividade do processo proposto se adaptam às atividades do processo orientado à reúso. Mais detalhes sobre as atividades do processo constam também no supracitado manual de uso e execução do processo no Apêndice A.

4.1.1.1 Especificação de requisitos

O modelo de processo orientado à reúso inicia com a especificação dos requisitos e no processo proposto optou-se por dividir essa atividade em duas: (i) Especificação dos estados de saúde que se deseja monitorar, bem como dos sensores e dos tipos de padrões de movimento que serão analisados com base nos dados coletados pelos sensores; e a (ii) Especificação dos demais requisitos.

A primeira atividade define o ponto principal da aplicação que deve ser desenvolvida, que são os estados de saúde a serem monitorados. É provável que dados os sensores que serão utilizados, mais estados de saúde possam ser detectados, porém, é preciso primeiro definir qual o foco principal da aplicação, que no caso de uma aplicação IoHT, envolve diretamente monitorar estados de saúde, seja para prover um melhor controle da saúde e consequentemente melhorar a qualidade de vida de um paciente, seja para detectar possíveis situações de risco, ou para auxiliar na reabilitação de pacientes, ou ainda simplesmente para prover um controle sobre atividades físicas e sinais vitais.

Guiado pelos estados de saúde definidos, são selecionados os sensores que serão utilizados, e os possíveis padrões de movimento que podem ser identificados pelos sensores. Vale ressaltar que essa atividade não é simples, pois muitos dos possíveis padrões de movimentação podem ser obtidos usando mais de um tipo de sensor, como o padrão de marcha, que, por exemplo, pode ser analisado usando dados de acelerômetros, giroscópios, ou dados de sensores de pressão. É fundamental que os analistas de requisitos definam bem que tipos de padrões de movimentação e sensores serão utilizados, uma vez que isso irá impactar diretamente no funcionamento da aplicação. Nesse sentido, artefatos que relacionem sensores e estados de saúde, como modelos, catálogos, ou taxonomias, podem ser úteis para guiar a tomada de decisão necessária a ser feita nessa primeira atividade do processo.

Na segunda atividade é feita a elicitação dos demais requisitos específicos da apli-

cação solicitada pelo cliente, levando em conta as regras de negócio. Nessa atividade pode ser utilizado qualquer processo de levantamento de requisitos conhecidos pelo time de desenvolvimento, incluindo os processos de elicitação de requisitos apresentados em (SOMMERVILLE, 2010; PRESSMAN; MAXIM, 2016).

4.1.1.2 Análise de componentes e alterações nos requisitos

A atividade de Análise de Componentes do modelo de processo orientado à reúso foi levemente adaptada para considerar o *loop* de adaptação (*MAPE-k Loop*), o que significa que durante a análise dos componentes deve ser considerado que os componentes possam contribuir para o ciclo de execução contínuo da aplicação.

Uma vez que os requisitos iniciais foram especificados e os componentes que serão reusados foram analisados e selecionados, é necessário que seja feita a alteração dos requisitos para se adequarem aos componentes. Conforme os requisitos são atualizados, o time envolvido também pode precisar atualizar casos de uso ou estórias de usuários. Também nesta atividade de processo, deve ser gerada ou atualizada a lista das possíveis adaptações da aplicação IoHT autoadaptativa. Além disso, caso o time de desenvolvimento esteja interessado em usar um método ágil iterativo, por exemplo, o SCRUM, esta atividade pode ser vista como o primeiro passo da iteração, e nela o time deve selecionar os casos de uso ou estórias de usuários que serão implementados em cada iteração.

4.1.1.3 Projeto com Reúso, Implementação e Integração e Validação e Verificação do sistema em tempo de projeto

Para o processo proposto, a atividade de Projeto com Reúso foi dividida em duas outras: (i) Criação das regras de adaptação e (ii) Projeto com reúso considerando os componentes do *loop* MAPE-K.

Primeiro é feita a criação das regras de adaptação de acordo com a lista de adaptações definidas na etapa anterior. Durante essa atividade é fundamental que sejam definidas quais são as possíveis alterações no contexto da aplicação devem ser consideradas para adaptação, incluindo as mudanças identificadas com base na entrada dos sensores e as possíveis alterações internas da aplicação. Nesse sentido é importante definir também como essa adaptação ocorrerá. Por exemplo, a alteração pode ser feita através da adição ou remoção de funcionalidades (como em (SANTOS *et al.*, 2017; JUNIOR *et al.*, 2018)), através do uso de atuadores (como em (SALEHIE;

TAHVILDARI, 2009; JUNIOR *et al.*, 2018)), ou através do uso da técnica de *Workarounds* (como em (ANDERSSON *et al.*, 2013)).

Em seguida é feito o projeto do sistema com reúso, considerando o *loop* MAPE-K. Os componentes reusáveis devem ser associados as diversas fases do *loop* de adaptação e deve ser tomada a decisão de como os componentes de cada fase irão se comunicar. É recomendado o uso de padrões de desenvolvimento de *software* como *Observer* (JUNIOR *et al.*, 2016; BARBOSA *et al.*, 2017), ou *Publish/Subscribe* (JUNIOR *et al.*, 2018) para troca de mensagens entre os componentes internos, e o uso de padrões adequados para comunicação entres dispositivos IoT ou entre dispositivos IoT e um servidor, como os padrões CoAP e MQTT (GÜNDOĞRAN *et al.*, 2018).

É fundamental que o motor de adaptação, o qual verifica as regras de adaptação, gera os planos de execução e dispara a execução desses planos, seja projetado de maneira que ele possa verificar as novas adaptações projetadas durante a atividade de evolução do *software*. Para isso é recomendado que seja definida uma linguagem comum para regras de adaptação que podem ser adicionadas ou removidas da lista de regras durante a fase de evolução, enquanto o sistema está executando. O uso de arquivos com uma lista de regras, como *templates* bem definidos (BUCCHIARONE *et al.*, 2017; JUNIOR *et al.*, 2016) ou programação orientada a aspectos (SANTOS *et al.*, 2017; SANTOS *et al.*, 2021) é recomendado para essa atividade.

Na atividade de Implementação e Integração, tanto no modelo de processo de desenvolvimento orientado à reúso, quando no processo MOTION, o sistema é codificado e os componentes e módulos são integrados.

Finalmente, como no modelo de processo orientado a reúso de componentes, após a implementação e integração do sistema, é necessário que o *software* seja validado. Para isso, é possível usar técnicas de teste, validação e avaliação de *software* conhecidas. Ao considerar o uso de uma metodologia iterativa junto com o processo MOTION, tal como o supracitado SCRUM, essa é a última etapa a ser executada em cada iteração e ao fim dessa atividade. Ainda havendo casos de uso ou estórias de usuários a serem desenvolvidos, volta-se à quarta atividade do processo. Caso contrário, o *software* pode ser entregue e utilizado pelo cliente. É importante que ao fim de cada iteração seja avaliada a qualidade da versão atual do *software*. Um processo de validação de sistema automatizado também deve ser planejado e desenvolvido durante a execução.

4.1.1.4 Evolução do Software e Validação e Verificação em tempo de execução

Uma vez que o sistema IoHT desenvolvido é autoadaptativo, durante sua execução é esperado que ele se adapte de acordo com as mudanças em seu contexto. Além disso, haja vista a natureza mutável dos *softwares*, novas adaptações podem ser adicionadas em tempo de execução. É importante também que a cada adaptação seja garantido que a integridade interna do *software* é mantida, bem como garantir que os requisitos funcionais e não funcionais continuam sendo atingidos. Nesse sentido, o processo MOTION considera duas atividades a serem executadas continuamente em tempo de execução, enquanto o *software* desenvolvido for utilizado.

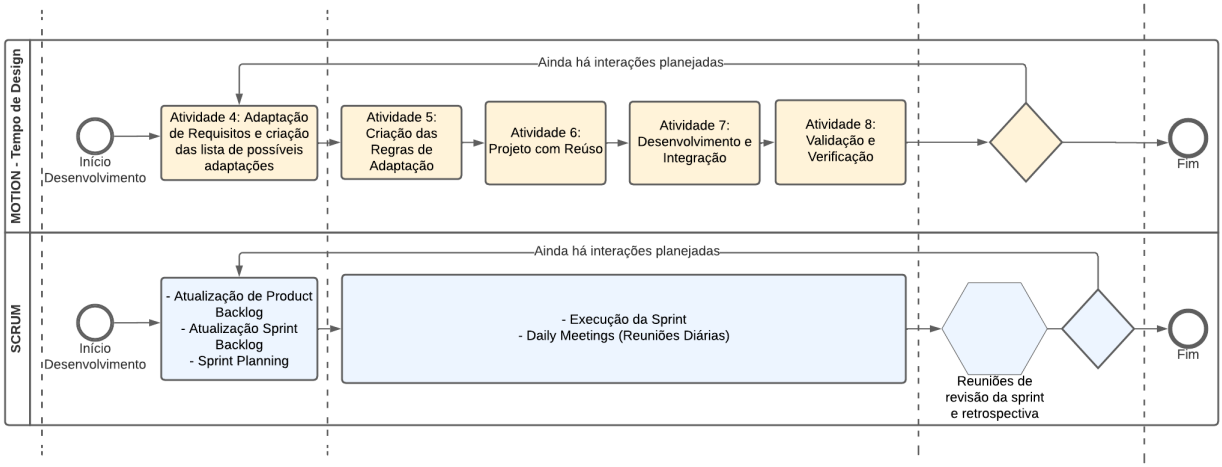
A atividade de Evolução do processo MOTION é a única que não apresenta correspondência direta com o modelo de processo orientado a reúso, mas é uma atividade considerada comum durante o desenvolvimento de *software* em geral. No processo MOTION, a Evolução consiste principalmente na criação de novas adaptações e regras de adaptação, ou remoção de regras de adaptação existentes. É importante notar que dependendo da maneira como o sistema é projetado, uma nova adaptação pode impactar pouco ou muito no funcionamento do *software*. Por exemplo, para um sistema cujas adaptações consistem no acionamento ou desligamento de *hardwares*, a adição de uma nova adaptação não deve impactar profundamente o comportamento interno ou a organização dos componentes do sistema (JUNIOR *et al.*, 2016). Por outro lado, para um sistema autoadaptativo baseados em serviços ou de funcionalidades, a adaptação pode acontecer pela adição ou remoção de serviços (BARBOSA *et al.*, 2017; JUNIOR *et al.*, 2016) ou pela adição e remoção de funcionalidades (SANTOS *et al.*, 2017). Nesse caso, uma adaptação pode ser capaz de alterar o comportamento interno do sistema.

Por fim, uma vez em execução, é importante que uma estratégia para Validação e Verificação do *software* durante a execução seja usada, pois é fundamental identificar se as adaptações sofridas pelo *software* impactam negativamente no seu funcionamento. Por esse motivo, a última atividade do Processo MOTION consiste na Validação e Verificação do *software* em tempo de execução. É importante ser cauteloso ao escolher a abordagem utilizada para esta atividade durante a execução, a fim de evitar *overhead*. Uma abordagem interessante para essa atividade, que pode ser usada com alguns sistemas autoadaptativos, é proposta em (SANTOS *et al.*, 2021)

4.1.2 Como Utilizar o SCRUM junto com o processo MOTION?

O processo MOTION foi desenvolvido para que pudesse ser utilizado junto a métodos em cascata ou incrementais em tempo de projeto, incluindo métodos ágeis como o SCRUM. A partir da Atividade 4 do processo MOTION é possível incorporar as atividades presentes no SCRUM adicionando os procedimentos relacionados ao desenvolvimento de sistemas autoadaptativos. Na Figura 16 é possível ver a relação das atividades do processo MOTION com as da metodologia SCRUM.

Figura 16 – Relação das atividades do MOTION com as atividade do SCRUM



Fonte: Próprio autor

Durante a atividade 4 do processo MOTION (Adaptação de Requisitos e criação da lista de possíveis adaptações) é executada a criação ou atualização do *Product Backlog* e do *Sprint Backlog*, além da reunião de planejamento da *sprint*. A *sprint* é executada durante as atividades 5 a 8 do processo MOTION. Por fim, ocorrem as reuniões de revisão e retrospectiva antes de ter início a próxima *sprint*.

4.2 Artefatos de reúso

A fim de prover mecanismos para auxiliar a execução das atividades do processo MOTION, também são propostos um conjunto de artefatos de *software*. Nesta Seção são apresentados os artefatos de reúso propostos nesse trabalho com o intuito de relacionar os dados de sensores a problemas de saúde e auxiliar a execução das atividades do processo proposto.

4.2.1 GRAFIT: Classification Graph for IoHT Applications

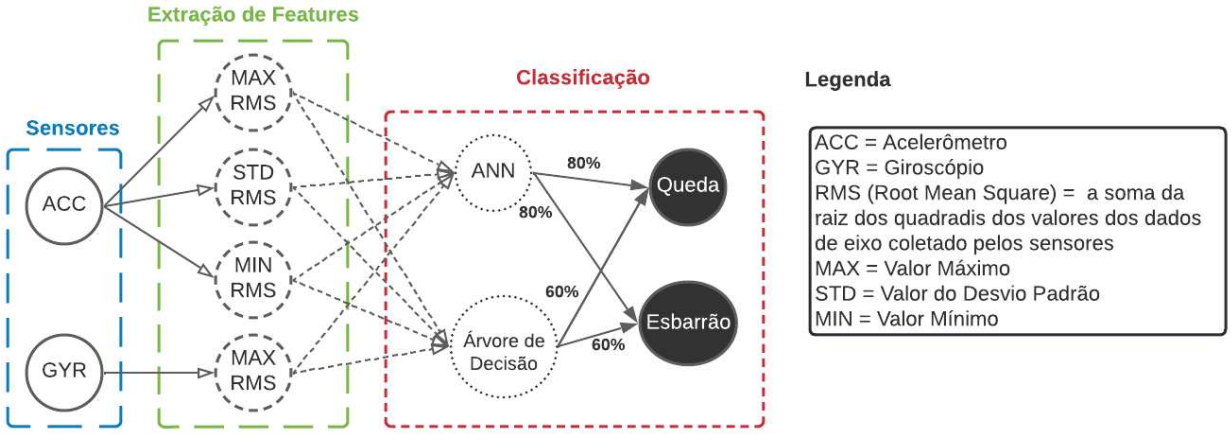
O *Classification Graph for IoHT Applications* (GRAFIT) é um artefato que visa auxiliar o time de desenvolvimento nas etapas de elicitação de requisitos, projeto, implementação de *software* e teste durante o uso de processos de desenvolvimento para construção de aplicações IoHT, inclusive o processo MOTION. Ele combina práticas comuns na literatura, para reconhecimento de atividades, situações de risco, ou problemas de saúde, a partir de dados coletadas de sensores pelos diversos sistemas IoHT baseados em dados de movimentação que foram identificadas durante o mapeamento sistemático da literatura descrito na Seção 3.

Este artefato modela, como uma estrutura de grafo, a relação entre sensores, *features*, algoritmos de classificação e situações ou estados de saúde. Além disso, esse modelo pode ser implementado como um artefato de *software* que pode ser usado como base para construir objetos de representação de conhecimento para aplicações IoHT. Os algoritmos de classificação são usados para prever as situações ou estados de saúde, com base nas *features* geradas a partir dos dados coletados pelos sensores. Esses algoritmos correspondem a algoritmos de aprendizado de máquina, algoritmos baseados em regras, algoritmos baseados em *thresholds*, algoritmos usados para clusterização ou qualquer algoritmo inteligente usado para classificar dados. Os estados ou situações de saúde podem ser compostos por doenças, atividades diárias, eventos de quedas ou qualquer informação utilizada para diagnosticar problemas de saúde ou situações de risco à saúde.

A Figura 17 apresenta um exemplo de grafo de classificação. Nesse exemplo, é possível ver os tipos de vértices e arestas que compõem o grafo. Os vértices representam os sensores, *features*, algoritmos de classificação e situações de saúde ou tipos de movimento. Já as arestas representam as relações entre os elementos representados pelos vértices.

A escolha de modelar o artefato proposto como um "grafo de classificação" aconteceu, principalmente, pela flexibilidade e facilidade de implementação de uma estrutura de dados em grafo, permitindo ao desenvolvedor escolher a forma que acredita ser a mais adequada para sua implementação. Quando essa estrutura de grafo é implementada, é possível a utilização de diversos algoritmos inteligentes para a análise de *features*, que são obtidas a partir dos dados coletados do paciente por meio dos sensores. Com base nessas *features*, é possível reconhecer padrões e classificá-los em estados finais que representam as situação de saúde do usuário do aplicativo. Além disso, o grafo de classificação proposto também pode ser otimizado para cada aplicação. Para isso, é necessário remover arestas indesejadas com base em regras no momento

Figura 17 – Exemplo de Grafo de Classificação



Fonte: Próprio autor

da geração do grafo para otimizá-lo para cada aplicação. Por exemplo, podem ser considerados apenas os caminhos no grafo entre os vértices que representam os sensores e os estados finais contemplando os sensores específicos utilizados pela aplicação que utilizará o grafo. Vários algoritmos existentes podem ser reaproveitados para exclusão ou poda de arestas para otimizar o uso do grafo, estratégia comumente utilizada para programação dinâmica.

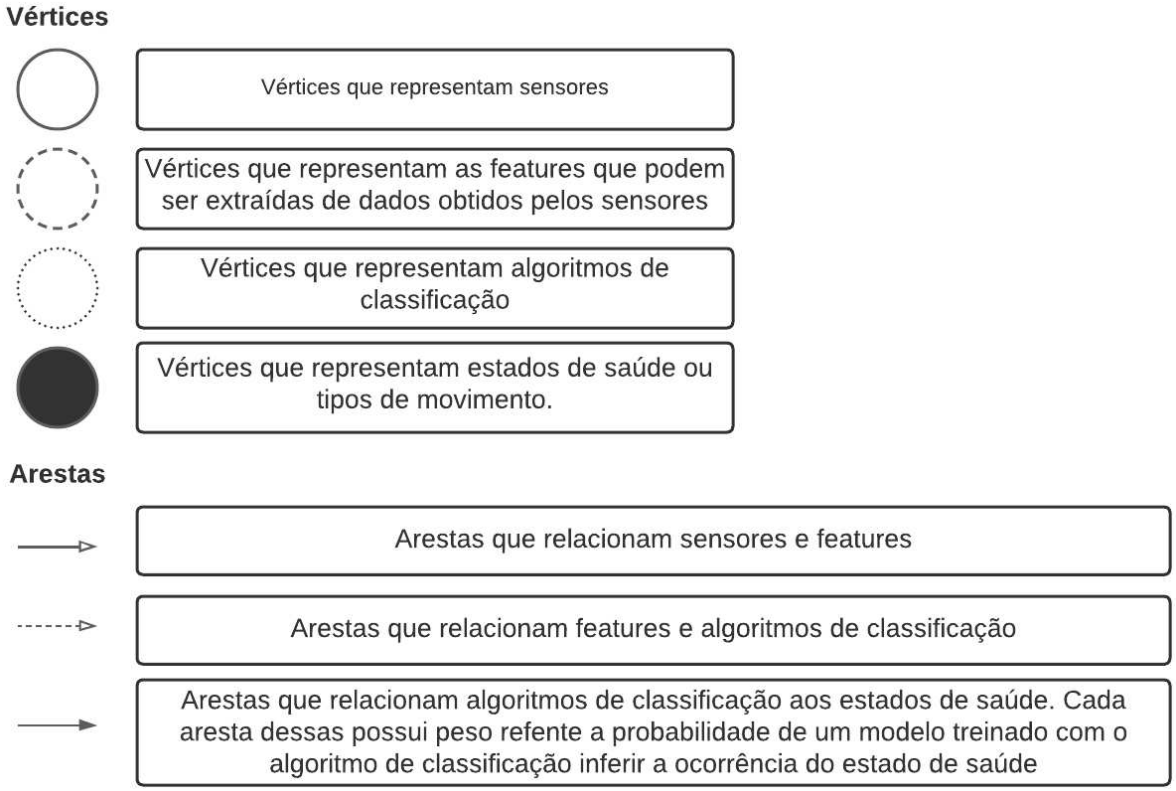
Os desenvolvedores podem usar um grafo de classificação proposto para ajudar a elicitação dos requisitos, já que o artefato fornece ao time de desenvolvimento exemplos de sensores, *features* e algoritmos de classificação que podem ser usados para desenvolver um aplicativo IoHT que visa identificar ou monitorar algumas das situações ou estados de saúde presentes nesse grafo de classificação. Além disso, arquitetos, analistas, desenvolvedores e testadores podem usar uma instância implementada do GRAFIT como base de conhecimento do aplicativo IoHT para analisar os dados obtidos pelos sensores e testar as funcionalidades dos aplicativos. Ainda é possível para o time de desenvolvimento construir uma base de conhecimento com seus experimentos para sua aplicação, utilizando como referência o modelo de grafo proposto neste trabalho.

4.2.1.1 Modelagem do GRAFIT

O GRAFIT é modelado como um grafo direcional composto por quatro tipos de vértices distintos que representam: sensores, *features*, algoritmos de classificação e situações de saúde ou tipos de movimentos. Existem também três tipos de arestas, dependendo dos tipos de vértices que elas vinculam. Em particular, as arestas que ligam os algoritmos de classificação as situações de saúde são ponderadas pela probabilidade do modelo inteligente treinado pelo

algoritmo de classificação inferir à situação de saúde. A Figura 18 apresenta os tipos de vértices e arestas que compõem o grafo.

Figura 18 – Elementos do GRAFIT



Fonte: Próprio autor

No exemplo da Figura 17 é apresentado um grafo de classificação com dois sensores, um sensor de acelerômetro e um sensor giroscópio, e quatro *features*, uma baseada nos dados do giroscópio e três baseadas nos dados do acelerômetro. As *features* são usados como entrada para dois algoritmos inteligentes no exemplo: um algoritmo de rede neural artificial e um algoritmo de árvore de decisão. Por fim, com certa probabilidade, os algoritmos classificam os padrões de movimento reconhecidos em dois estados que representam possíveis riscos à saúde: queda e esbarrão. Essas probabilidades podem ser obtidas avaliando os modelos inteligentes derivados de algoritmos já treinados usando um conjunto de dados de teste.

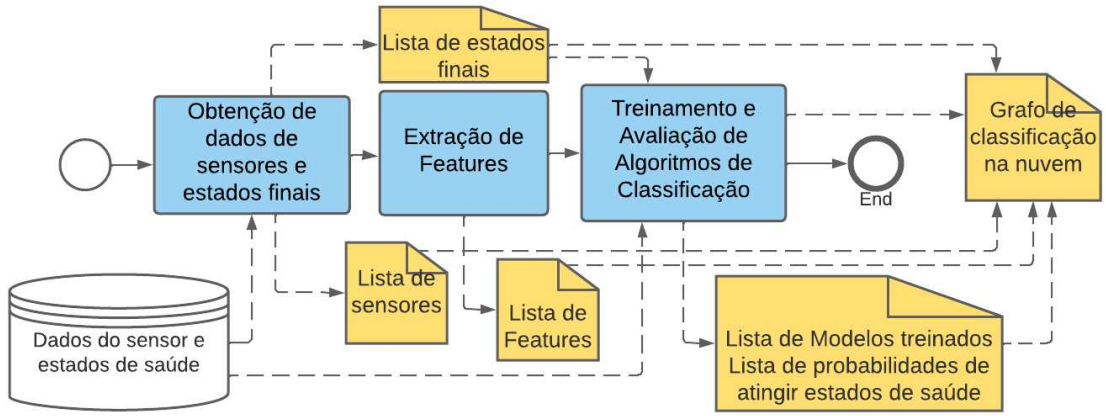
É possível usar diferentes algoritmos de classificação dependendo das *features* que representam as entradas desses algoritmos e dos estados de saúde que se deseja alcançar. Alguns algoritmos de classificação que podem ser usados incluem algoritmos baseados em *thresholds*, algoritmos baseados em regras, redes neurais artificiais e algoritmos baseados em árvore.

4.2.1.2 Construção do GRAFIT

Cada grafo de classificação é construído com base em uma ou mais bases de dados. As bases de dados usadas devem ter pelo menos uma lista de sensores, os dados coletados pelos sensores e as situações de saúde relacionadas aos dados coletados. Neste caso, o desenvolvedor do grafo de classificação precisa escolher o algoritmo de classificação e gerar as *features* relacionadas aos sensores. Os sensores e as situações de saúde usados para compor as bases de dados afetam diretamente o modelo de grafo e o tamanho a ser gerado, pois é a partir dessas informações que são definidos os vértices que irão compor o início e o fim de caminho dentro do grafo.

O processo ilustrado na Figura 19 é utilizado para gerar ou atualizar o grafo de classificação. Inicialmente é necessário obter bases de dados com dados de sensores relacionados à estados de saúde ou tipos de movimentos. Usando a base de dados, é possível extrair sensores, dados brutos dos sensores e estados de saúde ou tipos de movimento presentes na base de dados. Em seguida, o desenvolvedor do grafo de classificação precisa escolher as *features* relacionadas aos sensores e extrair os dados dessas *features* com base nos dados brutos dos sensores. No entanto, é possível usar *features* e dados de *features* da base de dados se essas informações estiverem disponíveis. Tomando como entrada as *features* extraídas, os algoritmos de classificação são treinados, resultando na geração de modelos treinados desses algoritmos e das probabilidades de atingir cada estado de saúde com base nos algoritmos treinados.

Figura 19 – Processo de criação ou atualização do GRAFIT



Fonte: Próprio autor

Sugere-se separar 10% dos dados da base de dados para avaliar o algoritmo de classificação, criar o modelo e obter as probabilidades, e os 90% restantes para treinar os

algoritmos. Por fim, o grafo é montado relacionando sensores, *features*, modelos inteligentes treinados usando os algoritmos de classificação e estados de saúde, destacando as probabilidades de atingir cada estado de saúde com base em cada modelo treinado.

Em relação ao tipo de medida usada para a probabilidade, a maioria dos estudos verificados durante o mapeamento sistemático, costuma considerar a probabilidade de acurácia para avaliar os modelos inteligentes. No entanto, dependendo da maneira como os dados são distribuídos na base de dados utilizada para construir o grafo, a probabilidade de outras medidas, como a precisão, podem ser mais confiáveis, uma vez que acurácia nem sempre é uma medida representativa em relação aos dados. Sendo assim, não é definido um tipo de medida específico para probabilidade de avaliação dos modelos inteligentes presentes no grafo de classificação, cabendo ao desenvolvedor do grafo identificar a melhor medida a ser utilizada.

Para utilizar mais de uma base de dados para atualizar ou criar um novo grafo, o desenvolvedor deve extrair os sensores das bases de dados, podendo agrupar sensores semelhantes de diferentes bases de dados como um mesmo elemento (vértice) no grafo. Também é necessário tratar os dados da base de dados e enviá-los em um formato padrão para o algoritmo que extrairá as *features*, considerando os tipos de sensores utilizados para coletar os dados. Uma vez que todos os dados tenham sido tratados, as etapas seguintes de criação do grafo são semelhantes a quando é utilizada apenas uma base de dados. Um exemplo de uso de duas bases de dados para construir um grafo de classificação é apresentado no Capítulo 5.

4.2.1.3 Usando uma instância do GRAFIT

Quando o grafo de classificação é construído, ele deve ser disponibilizado para uso pelos times de desenvolvimento de *software*. É sugerido que você disponibilize o grafo em um servidor para que seja possível que os usuários façam *download* tanto da versão completa quanto da versão otimizada do grafo de classificação. Os modelos inteligentes treinados pelos algoritmos de classificação também devem ser disponibilizados para *download*.

O grafo de classificação gerado pode ser armazenado em um banco de dados ou em um arquivo, e os modelos inteligentes treinados pelos algoritmos de classificação devem ser salvos em arquivos com os formatos necessários para que possam ser reutilizados pelas aplicações quando forem baixados.

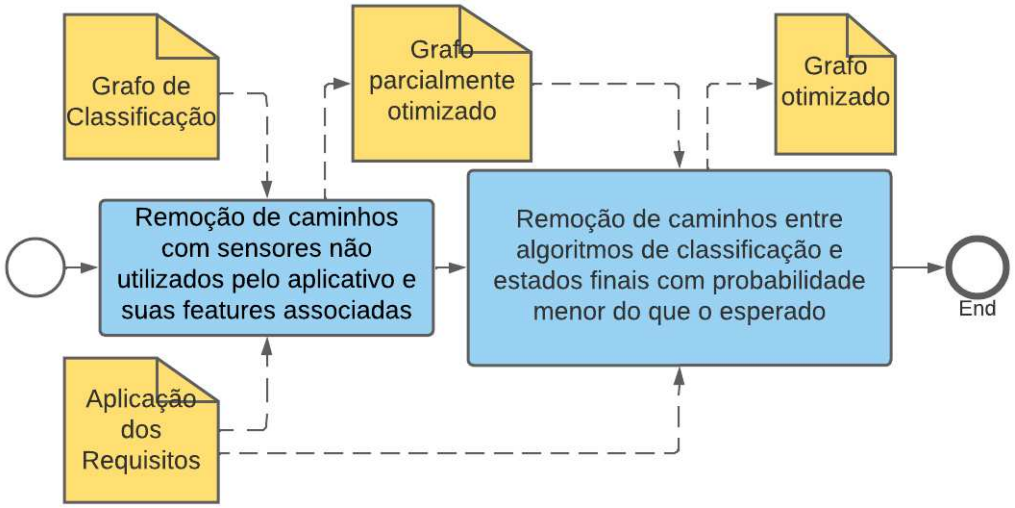
Após fazer o *download* do grafo de classificação e dos modelos treinados, a aplicação ainda precisará, com base nos dados coletados pelos sensores utilizados pela aplicação em

tempo real, extrair as *features* de acordo com o conjunto de *features* apresentados no grafo de classificação. Com as *features* extraídas, é possível utilizar os modelos treinados para inferir uma situação de saúde e assim planejar as ações a serem tomadas pela aplicação. É importante destacar que a aplicação deve utilizar as bibliotecas correspondentes àquelas utilizadas pelos modelos inteligentes treinados presentes no grafo de classificação para executar o processo de inferência e predição. Um exemplo de implementação do grafo de classificação pode ser visto no Capítulo 5.

4.2.1.4 Processo de Otimização do GRAFIT

Uma vez definidos os requisitos, o time de desenvolvimento pode optar por criar uma aplicação que não utilize todos os sensores do grafo de classificação. Nesse caso, não faz sentido usar o grafo completo, mas sim uma versão otimizada do grafo de classificação. Assim, é proposto um processo de otimização do grafo de classificação com base nos requisitos da aplicação, como pode ser visto na Figura 20.

Figura 20 – Processo de otimização do GRAFIT

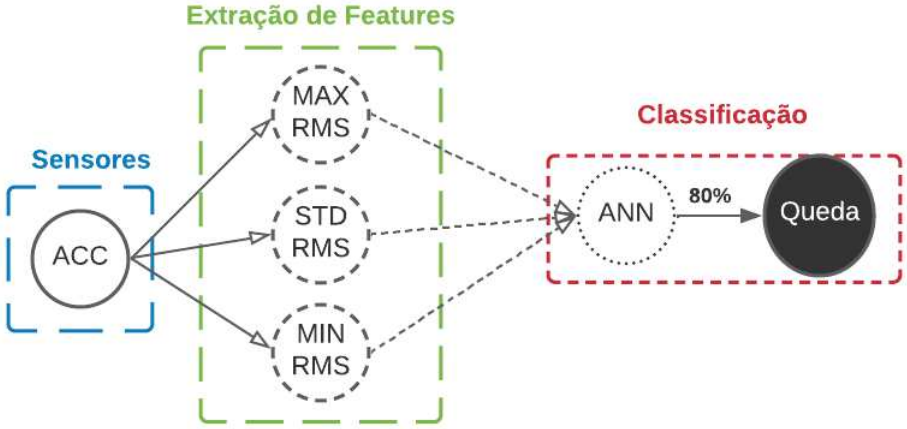


Fonte: Próprio autor

O processo de otimização começa com a aplicação solicitando o download do grafo otimizado. Recomenda-se o uso de uma API Web para lidar com a solicitação. Ao solicitar o grafo otimizado, o aplicativo informa o conjunto de sensores que utiliza e pode informar um valor mínimo de *threshold* relativo à probabilidade de acerto do modelo inteligente treinado com o conjunto de dados de teste utilizado durante a criação do grafo de classificação. A API Web recebe a requisição e executa o processo de otimização do grafo descrito na Figura 20,

removendo primeiro os caminhos do grafo associados aos sensores diferentes do solicitado. Essa remoção inclui os sensores não utilizados e todos os componentes, incluindo *features*, algoritmos de classificação com modelos treinados que usam as *features* dos sensores removidos como entrada e os estados finais (ou situações de saúde) que são alcançados apenas por esses modelos treinados. Com a primeira remoção de caminhos no grafo realizada, tem-se um grafo parcialmente otimizado.

Figura 21 – Exemplo de grafo otimizado



Fonte: Próprio autor

Em seguida, são removidos do grafo parcial os caminhos cujo peso de aresta entre os modelos treinados e os estados finais é menor que o *threshold* informado na requisição da aplicação. Como mostrado no exemplo da Figura 17, o peso dessas arestas corresponde à probabilidade de acerto alcançada após testar cada modelo dos algoritmos de classificação que foram treinados. Por fim, após a segunda etapa de remoção do caminho, obtém-se o grafo otimizado que será enviado ao aplicativo solicitante.

Um exemplo de uma otimização do grafo da Figura 17 pode ser visto na Figura 21. Nesse exemplo, a aplicação que solicitou o grafo otimizado utiliza apenas um sensor de acelerômetro e requisita apenas os modelos inteligentes cuja probabilidade de alcançar os estados finais é superior a 70%.

4.2.2 ARTe: Adaptation Rules Template

O objetivo do *Adaptation Rules Template* (ARTe) é fornecer um artefato que permita auxiliar a representação das regras de adaptação com base em um mecanismo simples de inferência, que verifica, dado um valor de contexto ou um conjunto de valores de contexto identificados, se uma ação específica deve ser executada pela aplicação. Porém, é importante

destacar que o *template* ARTe não indica como as ações ou os planos de ações devem ser executados, mas apenas provê um meio de auxiliar a identificação de que ação, ou plano de ação, deve ser executado, considerando cada regra especificada. O *template* ARTe utiliza arquivos em formatos ".xml", que podem ser compartilhados e editados enquanto a aplicação está em execução (similar ao que é citado em (BUCCHIARONE *et al.*, 2017; JUNIOR *et al.*, 2016)). A versão atual do *template* ARTe está disponível em: https://bit.ly/3pCvdzl.

A Figura 22 apresenta um exemplo do *template* contemplando os elementos usados para sua manipulação. O arquivo contém uma lista de regras de adaptações (`<adaptationRules>`). Para cada elemento (`<adaptationList>`), há uma lista de elementos de adaptação (`<adaptation>`). Para cada adaptação existe uma ação ou plano de ação (`<action>`) e uma lista de contextos a serem verificados (`<contextList>`). Para cada contexto (`<context>`) em um lista existe um identificador do contexto (`<name>`), um sinal (`<signal>`, que identifica o tipo de verificação a ser feita no contexto, e o valor *threshold* (`<value>`), ou lista de valores, que o contexto deve assumir para que a ação seja executada.

Figura 22 – *Template* ARTe

```
<adaptationRules>
  <adaptationList>
    <adaptation>
      <action>idAction1</action>
      <contextList>
        <context>
          <name>idContext1</name>
          <signal>equal</signal>
          <value>idContext1Threshold</value>
        </context>
        <context>
          <name>idContext2</name>
          <signal>greaterthan</signal>
          <value>idContext2Threshold</value>
        </context>
        <context>
          <name>idContext3</name>
          <signal>lessthan</signal>
          <value>idContext3Threshold</value>
        </context>
      </contextList>
    </adaptation>
  </adaptationList>
</adaptationRules>
```

(a) valores de contexto numéricos

```
<adaptationRules>
  <adaptationList>
    <adaptation>
      <action>idAction2</action>
      <contextList>
        <context>
          <name>idContext4</name>
          <signal>containValue</signal>
          <value>"value1"; "value2"; "value3"</value>
        </context>
      </contextList>
    </adaptation>
    <adaptation>
      <action>idAction3</action>
      <contextList>
        <context>
          <name>idContext5</name>
          <signal>notContainValue</signal>
          <value>"value1"; "value2"; "value3"</value>
        </context>
      </contextList>
    </adaptation>
  </adaptationList>
</adaptationRules>
```

(b) valores de contexto específicos

Fonte: Próprio Autor

No exemplo da Figura 22a, todos os tipos de contextos verificados são numéricos, de modo que a verificação identifica se o valor do contexto analisado (por exemplo, o nível de bateria, ou o valor de uma *feature* de movimentação) é "igual a" (valor de sinal *equal*), "maior que" (valor de sinal *greaterthan*), ou "menor que" (valor de sinal *lessthan*), o valor *threshold* especificado. Já na Figura 22b, os tipos de contextos verificados estão relacionados a

valores específicos “contidos” (valor de sinal *containValue*) ou “não contidos” (valor de sinal *notContainValue*) na lista de valores especificados.

Esse artefato pode ser usado para auxiliar as atividades de Criação das regras de Adaptação, Projeto com reúso e Implementação, presentes no processo MOTION. Além disso, uma vez que um arquivo “.xml” que contêm as regras de adaptação da aplicação com base no *template* ARTe é criado, recomenda-se que esse arquivo seja disponibilizado em um servidor, de modo que possa ser atualizado quando necessário sem que seja necessário modificar a aplicação. A aplicação deve então fazer o *download* automático da versão atual do arquivo de regras de adaptação quando for utilizá-lo.

4.2.3 KREATION: Kotlin Framework for Self-adaptive IoHT Applications

O *Kotlin Framework for Self-adaptive IoHT Applications* (KREATION) visa fornecer facilidades de código para auxiliar o desenvolvimento de aplicativos de IoHT autoadaptativos para dispositivos móveis com sistema operacional Android 12 ou superior. Para criar o *framework* KREATION foram combinados diferentes processos de desenvolvimento de *framework* encontrados na literatura (WILSON; WILSON, 1993) (YANG *et al.*, 1998) e foram adicionados padrões de *software* e artefatos reúso para auxiliar a implementação de aplicações IoHT autoadaptativas. O código do *framework* KREATION, sua documentação e manual de uso (em inglês) estão disponíveis em: https://bit.ly/3CQQ19j.

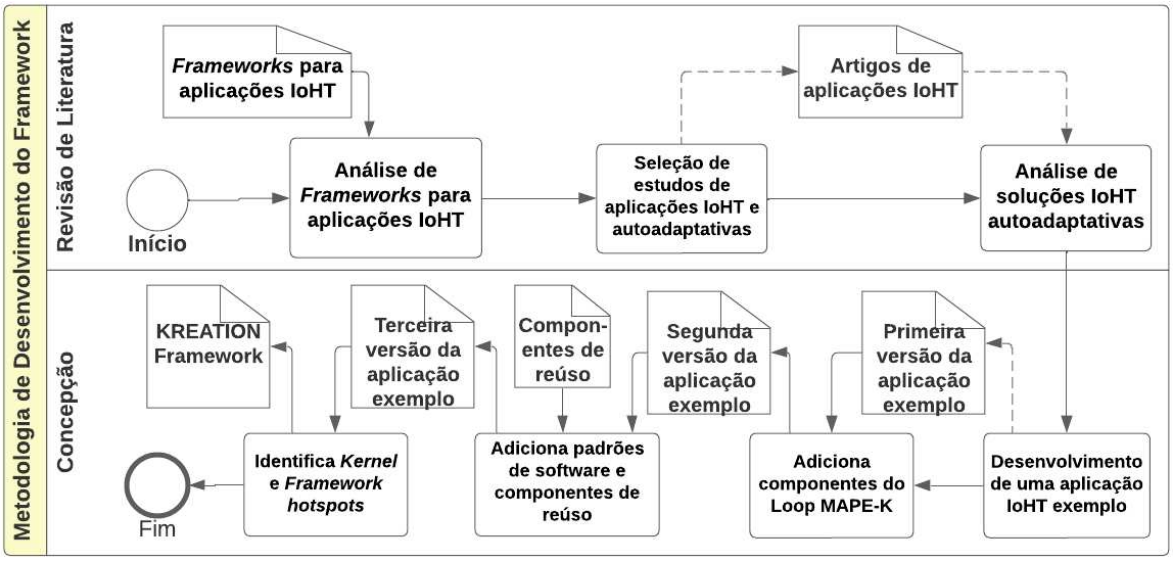
Para usar o *framework* KREATION, é precisa baixá-lo e abri-lo como um projeto na IDE Android Studio. Uma vez feito isso, o desenvolvedor pode então customizar o projeto para sua aplicação, modificando as partes que achar necessário nas classes *hot-spot* do *framework*. É possível também executar o projeto base do *framework* KREATION sem alterações para visualizar uma aplicação com testes das funcionalidades do mesmo. Esta aplicação base foi desenvolvida para executar em *smartphones* e permite a coleta de dados dos sensores do dispositivo e o uso da API Google Fit, mas devido às restrições desta API, para utilizá-la é necessário primeiro criar um identificador de aplicativo em uma conta no serviço de nuvem do Google.

4.2.3.1 Metodologia de Desenvolvimento do Framework

Para criar o *framework* KREATION, foram combinados dois métodos: o Processo de desenvolvimento baseado em Análise de Domínio e o processo de desenvolvimento baseado na Experiência com Aplicações (WILSON; WILSON, 1993) (YANG *et al.*, 1998). A Figura 23

ilustra o método utilizado para o desenvolvimento do *framework*.

Figura 23 – Metodologia usada para o desenvolvimento do *framework* KREATION



Fonte: Próprio autor

Primeiro foram analisados vários *frameworks* para sistemas IoHT (ARAÚJO, 2022) (AJERLA *et al.*, 2019) (SYED *et al.*, 2019)(BALAJI *et al.*, 2018) e aplicativos ioHT autoadaptativos e não autoadaptativos (BURGER *et al.*, 2020; KRUPITZER *et al.*, 2020; LENG *et al.*, 2020; EL-RASHIDY *et al.*, 2021). Após a análise desses estudos, foi desenvolvido um aplicativo exemplo que contém os principais elementos que os aplicativos IoHT para dispositivos móveis costumam possuir. Além disso, foram incluídos os elementos essenciais para que o aplicativo seja autoadaptativo com a implementação do mecanismo que permite que o aplicativo analise seu contexto, que é captado pelos sensores do dispositivo, e se adapte automaticamente a ele. A aplicação desenvolvida, bem como o *framework* KREATION, implementa módulos para cada fase do *loop* de adaptação MAPE-K. Essa aplicação faz parte da primeira prova de conceito dessa pesquisa que é apresentada no Capítulo 5.

Em seguida, foram incorporados padrões de *software* e artefatos de reúso que facilitam o desenvolvimento de componentes para este tipo de aplicação, conforme proposto em (PREE, 1995). A arquitetura do *framework* KREATION segue o padrão *Model View Control* (MVC) para arquitetura geral de aplicativos. Além disso, para auxiliar a comunicação e manipulação de componentes de *software*, foi utilizado o padrão *Observer*, que também é usado em (JUNIOR *et al.*, 2016), para comunicação entre os módulos referentes ao *loop* MAPE-K, e os padrões *Data Access Object* (DAO) e *Repository*, para facilitar o acesso e gerenciamento do banco de dados interno do dispositivo. Também foram incorporados elementos de outros

artefatos de reúso, como SUCCEEd (JUNIOR *et al.*, 2018), que ajudam a orquestrar a execução das ações em sistemas autoadaptativos.

Finalmente, com base na aplicação desenvolvida, foram identificados os componentes que compõem o *kernel* e os *hotspots* (PREE, 2000) do *framework* KREATION. O *kernel* contém os componentes que não devem ser alterados em um *framework*. Por outro lado, os *hotspots* são os elementos personalizáveis, as classes que podem ser alteradas e as interfaces e classes abstratas que podem ser estendidas. Para finalizar o *framework*, também foram identificados quais componentes de *software* são essenciais para o desenvolvimento de aplicações, mas que não fazem parte nem do *kernel* nem dos *hotspots* do framework, por exemplo, as telas dos aplicativos.

4.2.3.2 Componentes do Framework KREATION

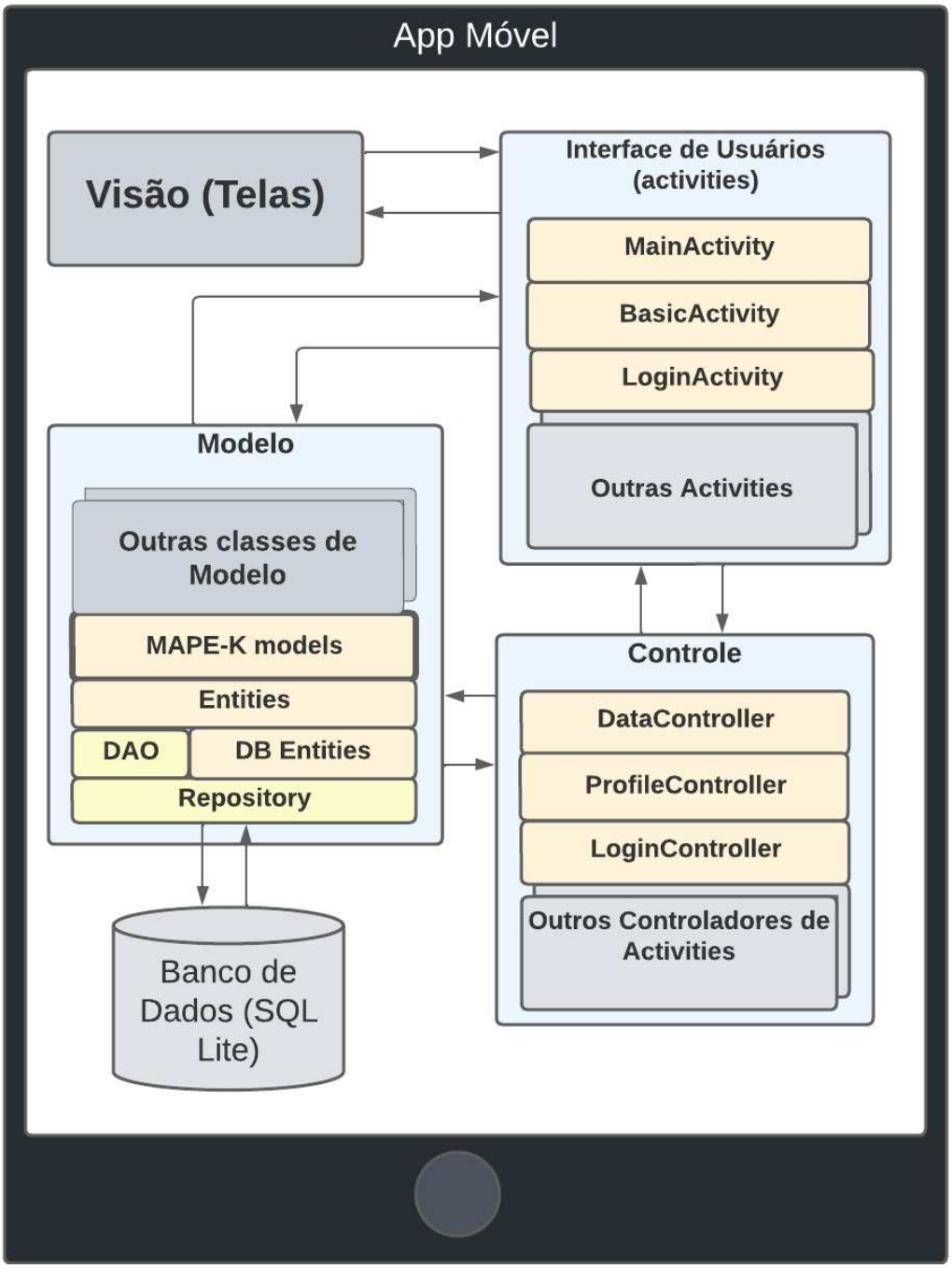
A Figura 24 mostra os principais componentes do *framework* KREATION. Como supracitado, foi utilizado o padrão MVC para compor sua arquitetura, dividindo e adaptando a camada de *View* em duas camadas, sendo uma a camada de Visão, composta pelas telas da aplicação, e a outra contendo as *Activities* e *Fragments* que controlam o funcionamento das telas e seus componentes. Na Figura 24, os componentes em laranja e amarelo contém as classes do *kernel* do *framework* e os *hot-spots* que podem ser customizados para cada aplicação. Os elementos em cinza são específicos de cada aplicativo e devem ser desenvolvidos pelo time de desenvolvimento.

O manual do usuário do *framework* KREATION indica quais componentes são *hot-spots*, aqueles que devem ser alterados pelo time de desenvolvimento, e quais componentes pertencem ao *kernel*, aqueles que não devem ser alterados. No código do *framework* também há comentários que indicam quais partes são *hotspots* e quais são partes do *kernel*.

Visão e *Activities*

Como já mencionado, o elemento *View* foi dividido em dois componentes. O primeiro componente inclui as telas do aplicativo, e o outro inclui as *Activities* e *Fragments*. As telas em um aplicativo Android são incorporadas ao aplicativo como uma série de arquivos ".xml" que definem os elementos gráficos e o design da tela. Esses componentes são interpretados diretamente pela plataforma Android. *Activities* e *Fragments* são classes utilizadas para controlar

Figura 24 – Principais componentes do *framework* KREATION



Fonte: Próprio autor

funcionalidades relacionadas as telas ou *Fragments* (*containers* ou *frames* que podem compor partes da tela). As *activities* herdam recursos de classes especiais do Android e obtêm o contexto do aplicativo em execução, o que permite controlar os elementos visuais das telas e acessar os recursos do dispositivo móvel.

O *framework* KREATION não contempla elementos específicos para construção de telas, pois as telas devem ser construídas exclusivamente para cada aplicação. Cabe aos *UX/Designers* e demais membros do time de desenvolvimento da aplicação definir os elementos

visuais. A única restrição do *framework* é a exigência de que o aplicativo possua pelo menos uma tela principal para iniciar a coleta de dados do sensor e o *loop* de adaptação, vinculando assim esta tela à classe `MainActivity.kt`. Esta tela não precisa ser a primeira tela do aplicativo. É sugerido que, em alguns casos, seja implementada uma tela anterior a essa nas aplicações para o usuário do aplicativo efetuar o *login*.

O *framework* KREATION mantém a classe `BaseActivity.kt` em seu *kernel* referente as *Activities*. A classe `BaseActivity.kt` não está associada a nenhuma tela específica, mas contém elementos (atributos e métodos) importantes para o uso do *framework*. Ela serve como um objeto de contexto para as classes de modelo que precisam ser ligadas a um elemento *Activity* para acessar os dados do dispositivo. Além disso, essa classe é herdada pela classe `MainActivity.kt`, e seus métodos podem ser usados para instanciar e iniciar as etapas necessárias para o funcionamento da aplicação a ser desenvolvida com o *framework*.

A `MainActivity.kt` é a classe *hotspot* principal do componente *View*. Ela está associada à tela do aplicativo que inicia o processo de execução do *loop* de adaptação. Esse processo deve ser iniciado em uma *thread* para garantir que seja executado em segundo plano. Vários componentes comuns não precisam ser alterados, mas o desenvolvedor deve customizar os elementos da `MainActivity.kt` que ele acredita serem necessários para a aplicação, em especial, os componentes de controle dos elementos visuais da tela.

O *framework* KREATION também possui uma classe *activity* opcional e *hotspot* para a tela de *login*. A classe `LoginActivity.kt` inclui métodos de *login* padrões usando a conta de usuário do Google.

Vale ressaltar que cabe ao time de desenvolvimento a criação de classes *Activities* ou *Fragments* para controlar os elementos visuais das demais telas projetadas, caso a aplicação Android tenha sido projetado para ter mais de uma tela.

Controle

As classes de Controle nas arquiteturas MVC controlam, processam e manipulam dados coletados ou exibidos pelos elementos da camada de Visão (as telas no caso de aplicativos móveis). Em aplicativos Android, a parte de controle dos elementos visuais geralmente é feita pelas classes *Activities* e *Fragments*. Em contraste, as classes de controle podem tratar e manipular os dados coletados por essas classes dados.

Os desenvolvedores podem criar classes de controle para cada *Activity*, mas o *framework* KREATION fornece algumas classes de controle *hotspot*. Uma classe para controle de login do usuário que se relaciona diretamente com a classe `LoginActivity.kt`. Uma classe para controlar os dados do perfil da conta do Google do usuário. E uma classe para controlar dados coletados da plataforma Google Fit, que pode ser usada para manipular dados coletados por diversos dispositivos, como *smartwatches* e *smartbands*. Também foi disponibilizado no *framework* KREATION uma classe para auxiliar no controle e atualização dos lançamentos de diferentes versões do aplicativo, caso desejado.

Modelo

Os componentes do Modelo são responsáveis pela lógica interna do aplicativo e geralmente são instanciados por classes *Activity* que acionam sua execução. E retornam as saídas de sua execução para uma outra classe Modelo, caso não seja um resultado a ser apresentado ao usuário, ou para uma classe de Controle, que posteriormente apresenta essa informação para uma classe *Activity* ou *Fragment*, ou ainda pode retornar diretamente a uma classe *Activity*.

As classes Modelo criadas para o *framework* KREATION foram divididas em módulos e submódulos e incluem as Entidades que representam as informações dentro da aplicação, os módulos que contém as classes referentes ao *loop* de adaptação MAPE-K, classes para controle do banco de dados SQLite interno ao dispositivo, classes para acessar APIs de terceiros disponíveis em um servidor externo, como um servidor de nuvem, e classes utilitárias, que contêm funções gerais do aplicativo.

Modelo - Classes de Entidade

As classes de Entidade representam objetos responsáveis por conter dados relevantes e que são manipulados por outras classes na aplicação. Esses objetos geralmente têm apenas atributos, e métodos *getters* e *setters*. As principais entidades contêm: (i) a representação da Base de Conhecimento (`KnowledgeRepresentation.kt`) e seus componentes; (ii) a representação das regras de adaptação; (iii) a representação dos dados que compõem o contexto do aplicativo (por exemplo, dados de recursos ou nível da bateria); e (iv) a representação dos dados coletados

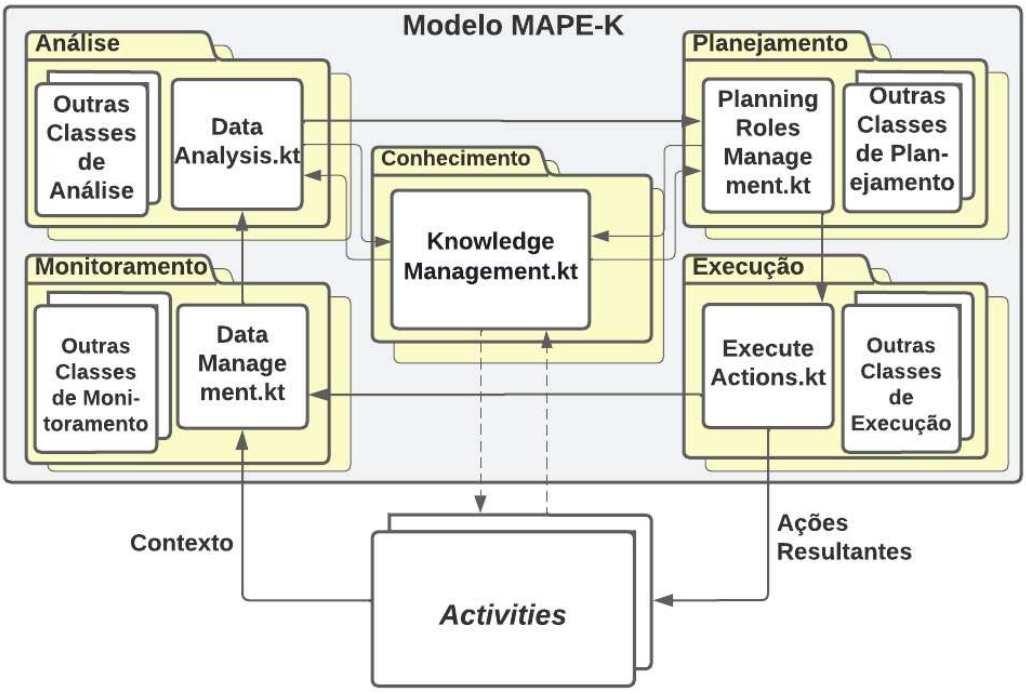
por um sensor.

A entidade `KnowledgeRepresentation.kt` foi construída com base no modelo GRAFIT apresentado anteriormente, para representar a base de conhecimento. Para cada tipo de vértice e aresta há uma classe entidade associada a ela no módulo de Entidades.

Modelo - Módulos do loop MAPE-K

A Figura 25 mostra como os módulos do *loop* MAPE-K se relacionam dentro do *framework*. Observe que tanto o contexto monitorado quanto o resultado das ações estão vinculados às *activities*, que são as classes que controlam os elementos visuais dentro de um aplicativo Android e, portanto, também são elas que podem solicitar acesso diretamente aos dados dos sensores dos dispositivos ou APIs do Google. Todas as classes e interfaces dos módulos do *loop* MAPE-K, exceto as interfaces *Observers*, são *hot-spots*.

Figura 25 – Esquema de implementação do *loop* MAPE-K



Fonte: Próprio autor

No *framework* KREATION, cada um dos módulos que se refere respectivamente às etapas de Monitoramento, Análise, Planejamento e Execução do *loop* MAPE-K possui uma classe principal, que é responsável por controlar as atividades realizadas em cada etapa do *loop* de adaptação. Além disso, cada um desses módulos também possui outras classes responsáveis

por funções específicas dentro de cada etapa do *loop* MAPE-K. O módulo responsável pela gestão do conhecimento também possui uma classe para manipular a base de conhecimento (um objeto `KnowledgeRepresentaion`). Além disso, conforme ilustrado na Figura 25, a base de conhecimento e a classe de gerenciamento da base de conhecimento também podem ser instanciadas e usadas pela classe `MainActivity.kt`. O time de desenvolvimento pode optar por chamar o método de criação do objeto `KnowledgeRepresentaion` na `MainActivity.kt` ou até mesmo deixar essa responsabilidade para a classe principal do módulo de Análise na primeira execução do *loop* de adaptação.

O módulo de monitoramento possui a classe principal`DataManagent.kt`, que é chamada uma vez pela classe `MainActivity.kt` para iniciar o *loop* de adaptação. Este processo deve executar em uma *thread* para que não precise ser iniciado novamente por uma classe *activity* durante a utilização do *app*, pois o *loop* de adaptação só deve parar de executar quando a aplicação for finalizada. A classe `DataManagent.kt` é responsável por receber os dados dos sensores, tratá-los e enviá-los para a etapa de análise. Quando o aplicativo precisa analisar um fluxo de dados, esta classe gerencia a janela de dados (que pode corresponder a um *buffer* com tamanho fixo ou a um período de tempo). Além dessa classe, o módulo de monitoramento contém classes e interfaces responsáveis por controlar a coleta de dados dos sensores e da API Google Fit. Para cada sensor do dispositivo, é recomendável criar uma classe de controle usando a interface `ICollectorData.kt`. Essa estrutura de coleta de dados dos sensores do dispositivo Android se baseia na mesma lógica empregada pelo *midleware* LoCCAM (MAIA *et al.*, 2013).

A classe principal do módulo de Análise é a `DataAnalysis.kt`. Esta classe é responsável por iniciar a extração de *features*, que é realizada por outra classe dentro do módulo de análise. Além disso, é responsável por analisar as *features* e obter os resultados que o módulo de planejamento utilizará, inclusive identificando mudanças de contexto que possam exigir planejamento de adaptações.

A classe principal do módulo de Planejamento é a `PlanningRolesManagement.kt`, que é responsável por gerar os planos de ação. Quando o módulo de análise identifica que houve uma mudança de contexto que requer adaptação, o plano de ação do aplicativo é alterado. Este módulo também contém uma classe para ler e interpretar um arquivos ".xml" com as regras de adaptação seguindo o *template* ARTe. O uso de um arquivo contendo regras de adaptação com base em um *template* específico permite que essas regras sejam alteradas mesmo com a aplicação em execução, caso seja de interesse do time de desenvolvimento. Sugere-se que este

arquivo de regras de adaptação seja disponibilizado em um servidor e que, de tempos em tempos, o aplicativo acesse este servidor para baixar a versão mais atualizada deste arquivo.

A classe `ExecuteActions.kt` é a principal do módulo de Execução. Esta classe é responsável por executar as ações planejadas. A lógica das ações são contidas em classes específicas para cada ação dentro do módulo de Execução. Essas classes herdam funções de uma mesma classe abstrata pertencente ao *framework* SUCCEEd (JUNIOR *et al.*, 2018), deste modo é possível usar uma abstração comum para todas ações, o que facilita a orquestração e priorização das ações a serem executadas.

A comunicação entre os módulos é feita usando o padrão de *software Observer* conforme proposto em (JUNIOR *et al.*, 2016). Para isso, a classe principal de cada módulo possui métodos para informar a classe principal do módulo da fase seguinte do *loop* de adaptação. Assim, ao finalizar as funções do módulo de monitoramento, a classe principal desse módulo (`DataManagent.kt`) informa a classe principal do módulo de análise (`DataAnalysis.kt`) para que a etapa de Análise seja executada. Ao final da execução da etapa de análise, a classe `DataAnalysis.kt` informa à classe `PlanningRolesManagement.kt` (classe principal do módulo de planejamento) que a etapa de Planejamento pode ser executada. De modo similar, ao final da etapa de planejamento, a classe `PlanningRolesManagement.kt` informa à classe `ExecuteActions.kt` (classe principal do módulo de execução) que as ações planejadas podem ser executadas. Por fim, ao final da execução das ações, a classe `ExecuteActions.kt` informa à classe `DataManagent.kt`, do módulo de monitoramento, que a execução terminou e o *loop* pode ser reiniciado.

O padrão *observer* permite maior desacoplamento entre cada módulo do *loop* MAPE-K, o que pode ajudar a dividir as tarefas de desenvolvimento de cada módulo entre diferentes programadores, além de ajudar a analisar e manter o funcionamento de cada módulo separadamente. Além disso, é possível desenvolver aplicações onde cada módulo de uma etapa do *loop* é observado por mais de uma instância do módulo da próxima etapa do *loop*. Assim, o desenvolvedor pode paralelizar ou dividir ações específicas na mesma etapa do *loop* de adaptação, se for de seu interesse. Por exemplo, é possível dividir a análise de dados em duas instâncias permitindo que um objeto que instancia a classe principal da etapa de monitoramento seja observado por dois ou mais objetos diferentes que instanciam a classe principal da etapa de análise.

O Módulo de Conhecimento possui apenas a classe `KnowledeManagement.kt`, que é responsável por construir e gerenciar o objeto *KowledgeRepresentation*, provendo métodos

também para que inferências e análises de dados possam ser feitas com base neste objeto. Por exemplo, é possível construir a representação de conhecimento com base no grafo de classificação apresentado anteriormente e identificar qual modelo treinado pelos algoritmos de classificação utilizados no grafo, possui maior probabilidade de identificar um estado final com base nas *features* extraídas dos dados coletados pelos sensores do dispositivo.

Modelo - Classes de Acesso a Servidores

Este módulo foi gerado para separar as classes responsáveis por acessar e manipular dados de servidores externos à aplicação, principalmente servidores em nuvem comumente utilizados em conjunto com aplicações móveis IoT. Por esta razão, este módulo no *framework* é chamado de *cloudConnection*. Recomenda-se que o time de desenvolvimento use esse módulo quando for preciso que aplicação se conecte ou requisite dados de outros sistemas em nuvem ou em qualquer servidor acessado via internet a partir do dispositivo móvel.

Foi fornecida uma classe *hotspot* opcional para *download* de instâncias do modelo de grafo de classificação GRAFIT através de requisição a uma API Web localizada em um servidor externo. A instância do grafo de classificação baixada pode ser utilizada para construir automaticamente um objeto *KnowledgeRepresentation* para a base de conhecimento da aplicação.

Modelo - Classes para Gerenciamento do Banco de Dados

Muitas vezes a aplicação móvel necessita manter um banco de dados relacional no dispositivo. Para aplicativos Android, é usado o banco de dados SQLite. Portanto, este módulo contém uma classe para manipulação de dados em um banco de dados SQLite. Para facilitar a manipulação de dados, foram usados os padrões de *software Data Access Object* (DAO) e *Repository*. O padrão *Repository* cria uma interface e um conjunto de métodos comuns para acessar os dados da tabela do banco de dados. O padrão DAO, por outro lado, é usado para criar classes para tratar as funções específicas de cada entidade do banco de dados. No *framewok* KREATION, as classes DAO herdam métodos de uma classe abstrata *Repository* e podem usá-los ou sobrescrevê-los.

As entidades do banco de dados estão presentes no submódulo *DBEntities* e correspondem às entidades que irão refletir as tabelas do banco de dados. Como exemplo, foi

disponibilizado junto com a aplicação base de *framework* uma entidade `Pessoa.kt` que representa as informações gerais de uma pessoa.

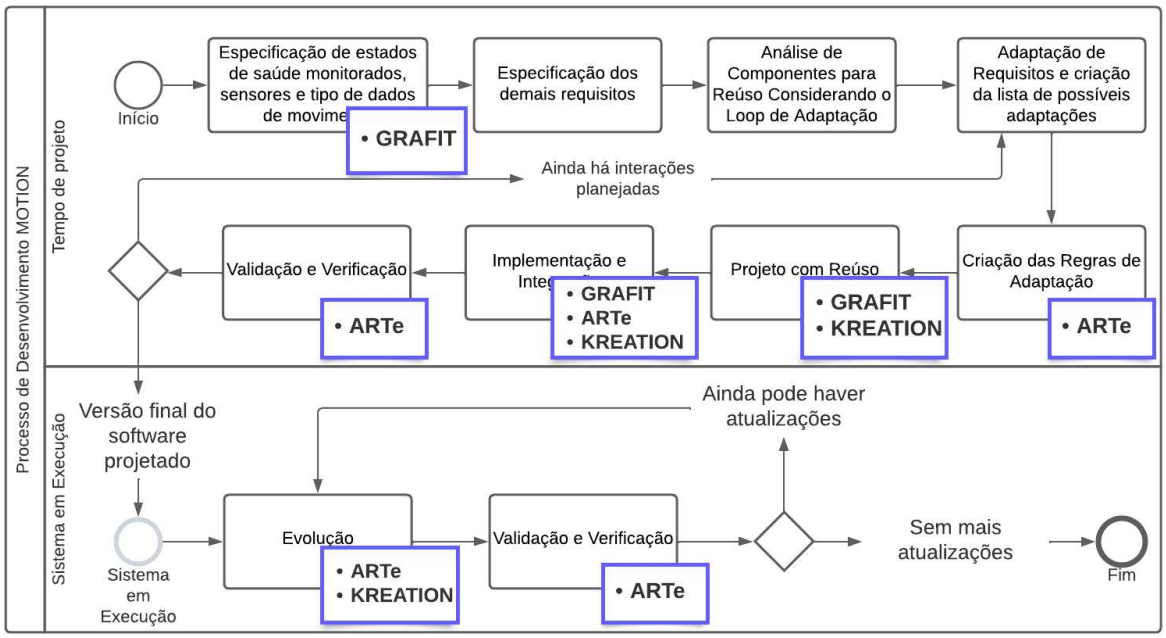
Modelo - Classes Utilitárias

Finalmente, este último módulo do Modelo contém um conjunto de classes utilitárias e objetos constantes que são úteis para o tratamento de diferentes objetos pelas outras classes do Modelo. Este módulo contém as principais constantes do aplicativo, métodos de criptografia e métodos de solicitação de permissão para acessar as APIs do Google e do *hardware* do dispositivo móvel.

4.2.4 Relação dos artefatos de reúso propostos com o processo MOTION

A Figura 26 apresenta quais atividades do processo MOTION podem ser auxiliadas pelos artefatos de reúso propostos.

Figura 26 – Relação de artefatos com o Processo MOTION



Fonte: Próprio autor

O modelo de grafo de classificação GRAFIT pode ser utilizado para auxiliar a escolha dos sensores que serão utilizados, com base nos estados de saúde que o sistema quer monitorar, já que ele fornece uma lista de sensores associados aos estados de saúde. Como ele também apresenta um lista de *features* e algoritmos de classificação também pode ser usados para auxiliar

a tomada de decisões durante o projeto da aplicação. Além disso, uma versão implementada do grafo pode ser utilizada para construção da base de conhecimento do sistema IoHT autoadaptativo e os modelos de algoritmos de classificação treinados podem ser aproveitados para inferência dos estados de saúde, auxiliando assim a implementação do sistema IoHT autoadaptativo.

Já o *template* ARTe pode ser usado para construir um arquivo contendo as regras de adaptação, que pode ser utilizado para implementar os mecanismos de tomada de decisão nas etapas de análise e planejamento do sistema IoHT autoadaptativo quando a adaptação ocorre, para planejar os testes e mecanismos de testagem do sistema, tanto em tempo de projeto quanto em tempo de execução, quando for necessário ocorrer uma adaptação. Além disso, você pode manipular arquivos construídos com base no *template* ARTe sem a necessidade de parar a execução do sistema, podendo inclusive adicionar e remover regras de adaptação, provendo uma auxílio na evolução do sistema IoHT autoadaptativo em tempo de execução.

Por fim, o *framework* KREATION apresenta uma série de funcionalidade e módulos, além de uma arquitetura própria, que pode ser utilizada para auxiliar o projeto do sistema IoHT autoadaptativo e a implementação e evolução desse software.

4.3 Conclusão

Este capítulo apresentou o processo de desenvolvimento MOTION proposto nesta tese. Também foram apresentados os artefatos de reúso de *software* propostos: o modelo de grafo de classificação GRAFIT, o *template* ARTe e o *framework* KREATION.

No próximo capítulo é descrito como o processo MOTION e os artefatos de reúso propostos foram avaliados.

5 PROVAS DE CONCEITO

Neste capítulo são apresentadas as provas de conceito (do inglês, Proof of Concept (PoC)) (HORTON; RADCLIFFE, 1995) utilizadas para avaliar a viabilidade e a funcionalidade do processo MOTION e dos artefatos de reúso propostos. Na Seção 5.1 é apresentada a primeira PoC, onde foi desenvolvida uma aplicação Android e um exemplo de implementação do grafo de classificação, desenvolvidos para avaliar a versão preliminar do processo MOTION e o GRAFIT, respectivamente. Duas outras aplicações desenvolvidas em Kotlin são apresentadas na Seção 5.2 como uma segunda PoC, com o intuito de avaliar a viabilidade e as funcionalidades do *framework* KREATION e do *template* ARTe. Por fim, na Seção 5.3 é apresentada a conclusão do capítulo.

5.1 Primeira prova de conceito

Nessa primeira PoC, foi desenvolvida uma aplicação Android em JAVA com o auxílio do processo MOTION. Também foi implementado um sistema de gerenciamento para criação, atualização e otimização de uma instância do GRAFIT usando a linguagem Python. O grafo de classificação gerado, em formato de um arquivo ".xml", bem como os modelos inteligentes treinados durante a geração do grafo foram disponibilizados para download via requisição a uma API Web, também desenvolvida em Python. O grafo de classificação foi utilizado como base de conhecimento pela aplicação Android e os modelos inteligentes treinados, que são apresentados no grafo, foram utilizados pela aplicação para inferência de diferentes tipos de movimento.

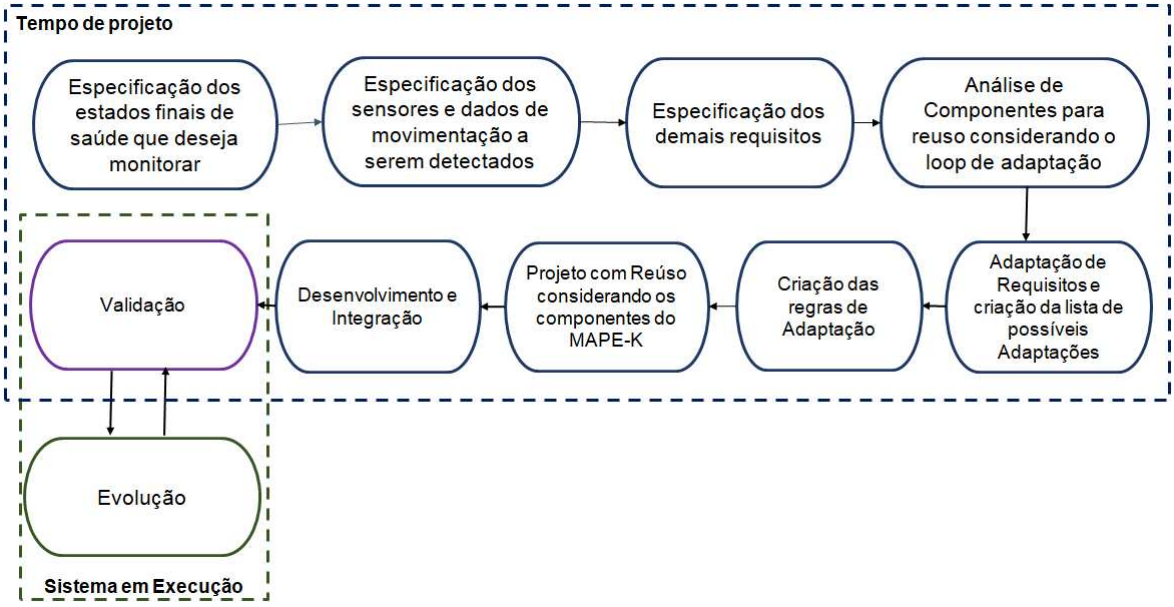
A aplicação Android teve como motivação a necessidade de avaliar se é possível construir uma pequena aplicação IoHT autoadaptativa usando a versão preliminar do processo MOTION (Figura 27), além de servir como base para extrair informações necessárias para construção do *framework* KREATION. Já a instância do grafo classificação foi implementada com o objetivo de avaliar a viabilidade e funcionalidade do grafo, além dos processos propostos para criação e atualização e para a otimização do grafo de classificação.

5.1.1 Aplicação Android da primeira PoC

Como citado previamente, a versão preliminar do processo MOTION foi utilizada para guiar o desenvolvimento da aplicação Android da primeira PoC. Na primeira atividade foi definido que a aplicação teria como finalidade identificar padrões de movimento que indiquem

a execução de atividades comuns e situações anômalas (quedas ou esbarrões). Para tal, na segunda atividade optou-se por utilizar apenas um sensor de acelerômetro. Essas duas atividades da versão preliminar do processo MOTION, foram unidas e se tornaram parte de uma mesma atividade na versão mais recente do processo.

Figura 27 – Versão preliminar do processo MOTION utilizada para o desenvolvimento da aplicação Android da primeira PoC



Fonte: Próprio autor

Após a escolha dos sensores, foram definidos os outros requisitos da aplicação, como a necessidade de avaliar o contexto do consumo de bateria, e que tipos de alertas e atuadores seriam utilizados quando uma situação anômala fosse detectada.

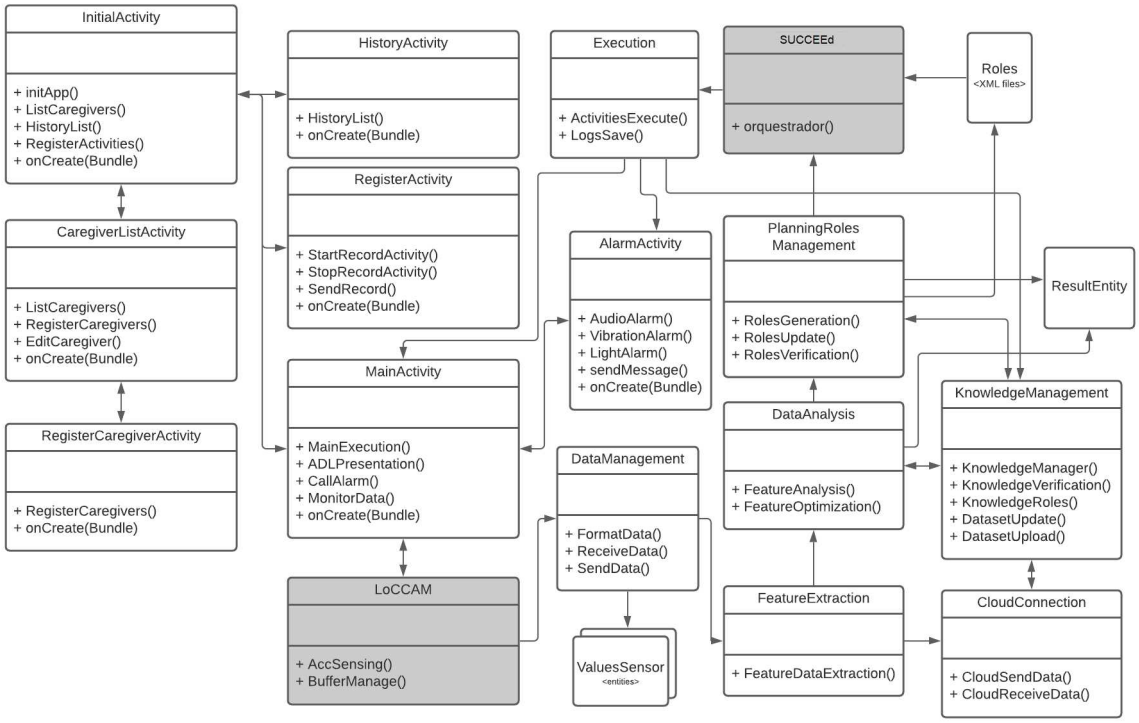
Com a elicitação inicial dos requisitos feita, foram analisados os componentes que poderiam ser reusados para auxiliar o desenvolvimento da aplicação. Entre os componentes selecionados, encontra-se o *middleware* LoCCAM (MAIA *et al.*, 2013), para aquisição de dados, e o *framework* SUCCEEd (JUNIOR *et al.*, 2018), para orquestração e execução de planos de ação. Foram então revisados os requisitos e gerada a lista de possíveis adaptações. A aplicação Android foi projetada para monitorar atividades e emitir alertas caso algumas atividades anômalas fossem identificadas. Esses alertas utilizam três recursos diferentes: texto, som e luz do *flash* da câmera do dispositivo. A adaptação projetada considera o contexto de bateria do dispositivo.

As regras de adaptação foram geradas e definidas em um arquivo ".xml" para possíveis edições durante a execução da aplicação. De acordo com as regras definidas, se o nível restante da bateria do dispositivo for superior a 70%, o aplicativo usará som, a luz do *flash* da

câmera e o texto no Alerta. Se o nível de bateria restante for maior que 40% e menor que 70%, o aplicativo usará a luz e o texto no Alerta. E se o nível de bateria restante for inferior a 40%, o aplicativo usará apenas textos no Alerta. O arquivo com as regras de adaptação foi colocado dentro de um diretório interno da própria aplicação.

Com os requisitos e as regras de adaptação definidos, a aplicação foi então projetada como uma aplicação Android nativa em Java usando a IDE Android Studio. A Figura 28 apresenta o diagrama de classes da aplicação, incluindo classes referentes ao *loop* de adaptação MAPE-K e também as `Activities`, que controlam as telas do *app*, segundo a própria nomenclatura utilizada para aplicações Android.

Figura 28 – Diagrama de classes da aplicação Android



Fonte: Próprio autor

A aplicação foi implementada seguindo como base o diagrama de classes apresentado na Figura 28. Posteriormente foram adicionadas novas classes de modelo, incluindo métodos para o download do grafo de classificação e dos modelos inteligentes treinados junto com o grafo. Usando a biblioteca TensorFlow Lite (DAVID *et al.*, 2021) foram gerados também um método para construção do objeto de representação de conhecimento com base no grafo de classificação e um outro método para inferência da atividade usando um dos modelos inteligentes. Por fim, a aplicação foi testada em um celular com sistema operacional Android versão 7. Durante a

execução da aplicação foi possível verificar a adaptação de acordo com a mudança do nível de bateria.

Por ser uma aplicação criada apenas com o intuito de avaliar de maneira preliminar o processo, não foi desenvolvida uma interface gráfica elaborada para a mesma. Foram usados apenas alguns elementos gráficos, como textos e botões, para demonstrar o funcionamento e inciar o ciclo de adaptação e inferência do tipo de movimento referente aos dados coletados pelo sensor de acelerômetro do celular utilizado para testar a aplicação. O código fonte da aplicação Android desenvolvida para a primeira PoC pode ser acessado em: https://bit.ly/3rutixd.

5.1.2 Implementação do Grafo de Classificação

Para avaliar a viabilidade do modelo GRAFIT, bem como dos processos de criação, atualização e otimização do GRAFIT apresentados no capítulo 4, foi implementado um sistema de gerenciamento para criação, atualização e otimização de uma instância do grafo de classificação. O sistema de gerenciamento do grafo de classificação foi disponibilizado em um servidor de nuvem localizado em uma instância EC2 da *Amazon Web Services* (AWS) (CLOUD, 2011). Além disso, foi desenvolvida e disponibilizada no mesmo servidor, uma API Web para tratar das solicitações de *download* do grafo otimizado em formato de um arquivo ".xml" e das solicitações de *download* dos modelos inteligentes que foram treinados durante a construção e atualização da instância do grafo de classificação gerada. O código do sistema de gerenciamento do grafo de classificação e da API Web estão disponíveis em: https://bit.ly/3p0xO5P.

5.1.2.1 Configurações

O sistema de gerenciamento do grafo de classificação e a API web foram implementados em Python 3. A API web foi implementada usando o *framework* Flask (DWYER *et al.*, 2017). A instância do grafo de classificação foi armazenada em memória em um banco de dados MYSQL (LUKASZEWSKI; REYNOLDS, 2010) e também foi disponibilizada em arquivo ".xml". Os algoritmos de classificação foram criados usando duas bibliotecas de aprendizado de máquina, o TensorFlow (DAVID *et al.*, 2021), para o algoritmo Rede Neural Artificial, e o Scikit-Learn (PEDREGOSA *et al.*, 2011), para os algoritmos de Árvore de Decisão e *Random Forest*.

O TensorFlow e Scikit-Learn são bibliotecas de aprendizado de máquina de código aberto. O TensorFlow é uma biblioteca de aprendizado profundo desenvolvida pelo Google e

é usada para uma variedade de tarefas, como reconhecimento de imagem, processamento de linguagem natural e aprendizado por reforço. Ela é construída sobre uma biblioteca chamada Theano e é altamente extensível. Já o Scikit-Learn (ou SKLearn) é uma biblioteca Python que fornece uma ampla variedade de algoritmos e ferramentas de aprendizado de máquina. Ela é usada para mineração e análise de dados e é comumente usada para aprendizado supervisionado e não supervisionado. O SkLearn é mais tradicional em sua abordagem, oferecendo suporte para vários algoritmos, como *Support Vector Machines*, Árvores de Decisão e *Random Forest*.

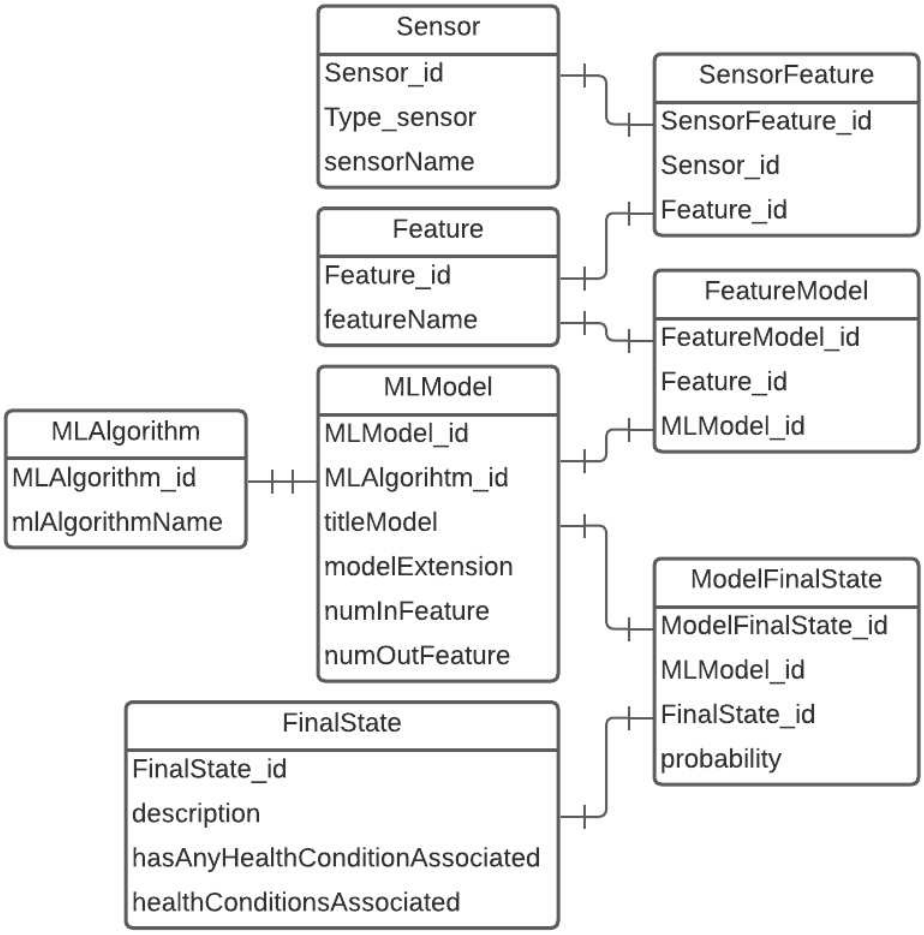
5.1.2.2 Implementação

Para criação e atualização do grafo de classificação, foi seguido o processo apresentado na Figura 19 e, para otimização, foi utilizado o processo da Figura 20. Ambos os processos foram detalhados no Capítulo 4 desse documento. Entretanto, antes de desenvolver as funcionalidades do sistema de gerenciamento do grafo de classificação, foram criadas entidades e um banco de dados para armazenar os dados do grafo de classificação e para geração dos arquivos ".xml" contendo o grafo de classificação completo e o grafo de classificação otimizado.

A Figura 29 apresenta o diagrama de entidades e relacionamentos dos componentes da instância do grafo de classificação. Como pode ser visto, há quatro entidades relacionadas aos vértices do grafo: Sensores (`Sensor`), *Features*(`Feature`), Modelos treinados usando os algoritmos de classificação (`MLModel`) e Estados Finais `FinalState` (correspondente as situações de saúde). Além disso, foi criada outra entidade, `MLAlgorithm`, para identificar especificamente o algoritmo de classificação usado para treinar cada modelo inteligente. As tabelas no banco de dados MySQL criado representam essas entidades. Além disso, também foram criadas três entidades relacionadas às arestas do grafo que representam as ligações entre os vértices: `SensorFeature`, `FeatureModel` e `ModelFinalState`.

A entidade `Sensor` contempla informações sobre diferentes sensores e tipos de sensores. O tipo de sensor `Type_sensor` corresponde a uma categoria de sensores, categorizando sensores semelhantes, por exemplo, dois modelos diferentes de acelerômetros com *hardware* bem diferente. No grafo, é utilizado o tipo de sensor para indicar o sensor. Em seguida, pode-se usar dados para diferentes sensores, mas com categorias semelhantes para gerar recursos e modelos de treinamento. A entidade `Feature` contempla informações sobre *features* geradas com base nos dados do sensor, por exemplo, uma média de dados coletados de um giroscópio. A entidade `MLModel` contempla o modelo de algoritmo de classificação treinado, geralmente

Figura 29 – Diagrama de Entidades



Fonte: Próprio autor

algoritmos de aprendizado de máquina. Essa entidade inclui o nome do modelo, o número de *features* usadas como entradas para o treinamento do algoritmo de classificação e o número de estados finais avaliados pelo algoritmo de classificação. A entidade `FinalState` contempla informações sobre as situações de saúde coletadas das bases de dados utilizadas para construção do grafo de classificação.

As entidades `SensorFeature`, `FeatureModel` e `ModelFinalState` correspondem à representação das arestas do grafo de classificação que representam o relacionamento entre sensores e *features*, entre *features* e modelos inteligentes treinados e entre modelos inteligentes treinados e estados finais. A entidade `ModelFinalState` inclui o valor da probabilidade de atingir o estado final utilizando o modelo inteligente treinado. Essa probabilidade é obtida com base na avaliação da base de dados de teste (composta por 10% dos dados da base de dados original).

Também foram criados métodos para popular o banco de dados com os dados referentes às entidades e um método para montar o arquivo ".xml" com as informações do grafo

de classificação a ser utilizado pelas aplicações. Um exemplo do grafo de classificação em um arquivo ".xml" está disponível em: bit.ly/3XYZHqC.

5.1.2.3 Pré-processamento

Com as entidades e o banco de dados criados, é necessário coletar os dados dos sensores e estados finais da base de dados e pré-processar os dados dos sensores para extrair as *features*.

Para a construção da instância do grafo de classificação, foram usadas duas bases de dados. Em ambos as bases de dados, os dados foram coletados em experimentos com diferentes voluntários que foram instruídos a realizar atividades usando dispositivos móveis contendo os sensores usados para coleta de dados. A primeira base de dados (B1), apresentada em (LINHARES *et al.*, 2020), é composta por conjuntos de *streaming* de dados de acelerômetro e giroscópio contemplando os seguintes movimentos: andar, bater em uma mesa, bater em uma parede, correr, deitar, esbarrão na parede, escrever, bater palmas em pé, bater palmas sentado, pular, queda para frente com apoio, queda para frente sem apoio, queda para os lados, sentar, sentar com apoio, sentar sem apoio e apalpar. Esta base de dados foi originalmente usada para treinar um aplicativo de detecção de queda de maneira binária, onde os movimentos de queda são parte de um conjunto (bit 1) e todos os demais movimentos como parte de um conjunto não queda (bit 0). A segunda base de dados (B2) é apresentada em (BRUNO *et al.*, 2013) e foi utilizada para analisar o reconhecimento de comportamento humano. Esta base de dados é composta por conjuntos de *streaming* de dados de acelerômetro contemplando as seguintes atividades diárias: escovar os dentes, pentear o cabelo, subir escadas, descer escadas, andar, beber de um copo, colocar água em um copo, comer com garfo e faca, comer com colher, usar o telefone, levantar da cama, deitar na cama, levantar de uma cadeira e sentar em uma cadeira.

Na maioria das vezes, as bases de dados de saúde contém os dados brutos dos sensores, mas não as *features* calculadas com base nesses dados. Se a base de dados contiver as *features*, pode-se coletar os sensores, *features* e estados finais diretamente dessa base de dados para, em seguida, treinar o modelo inteligente para geração dos dados necessários para criar o grafo de classificação. Caso contrário, é preciso executar um pré-processamento dos dados brutos dos sensores contidos na base de dados, para então extrair as *features*. Para essa prova de conceito, foi feito o pré-processamento e extração das *features* para ambas as bases de dados.

O pré-processamento de dados é específico para cada base de dados, portanto,

ao adicionar uma nova base de dados para atualizar o grafo de classificação, cabe ao desenvolvedor gerar primeiro um código específico para pré-processamento de dados. Durante o pré-processamento, os dados brutos são tratados e dispostos em uma estrutura padrão (geralmente os dados são dispostos em uma estrutura matricial). Uma vez que os dados estejam em uma estrutura padrão são aplicados os métodos de extração de *features*.

Nove *features* para os dados do sensor do acelerômetro de ambas as bases de dados foram extraídas e as mesmas nove *features* foram extraídas para os dados do giroscópio da base de dados B1. Assim, para a base de dados B1, foram extraídas 18 *features* (9 *features* com base nos dados do acelerômetro e outras 9 *features* com base nos dados do giroscópio), enquanto para a base de dados B2 foram extraídas 9 *features* com base nos dados de acelerômetro. Considerando apenas as *features* baseadas nos dados de acelerômetro é possível utilizar em conjunto as bases de dados B1 e B2, uma vez que os dispositivos contendo os sensores de acelerômetro usados para coleta dos dados são similares. As *features* extraídas dos dados do acelerômetro e do giroscópio foram: média, valor máximo, valor mínimo, desvio padrão, curtose, assimetria, entropia, desvio médio absoluto e intervalo interquartil. Os métodos da biblioteca de `statistics` do Python 3 foram utilizados para cálculo e extração das *features*.

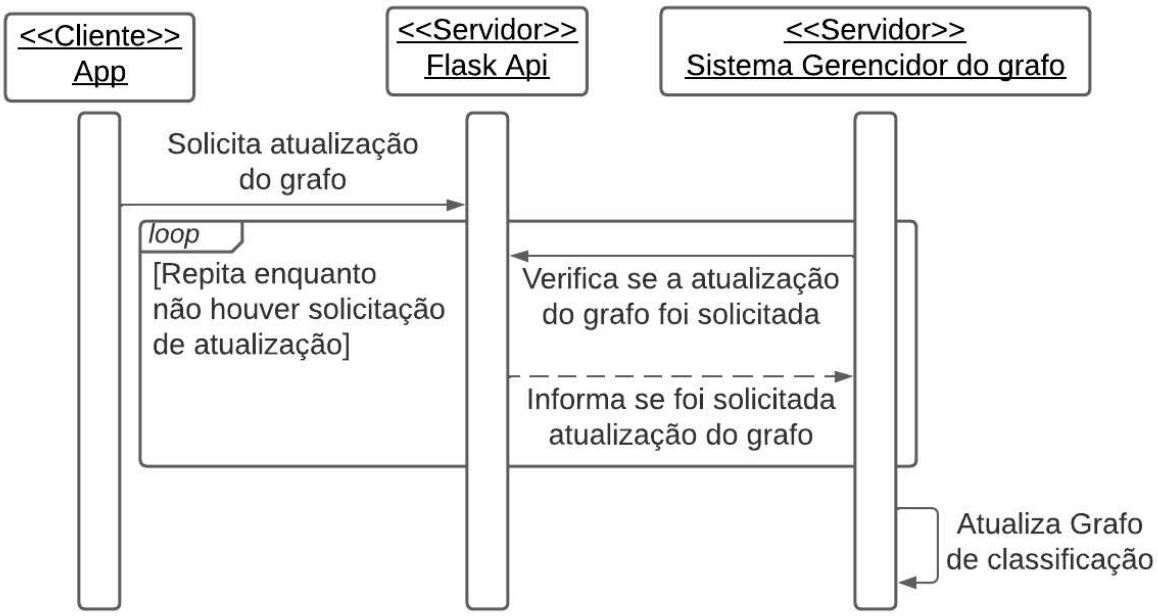
5.1.2.4 Criação ou atualização do grafo de classificação

O algoritmo de criação e atualização do grafo de classificação segue o processo descrito na Seção 4.2.1.2, e seus passos são ilustrados na Figura 30. Esse processo inicia com a aplicação, ou cliente, solicitando à API Web que seja feita a atualização do grafo de classificação. A API então informa ao sistema de gerenciamento do grafo no servidor que deve ser feita essa atualização. Caso não haja uma versão do grafo de classificação já criada, o sistema de gerenciamento do grafo irá gerá-la, caso contrário será feita a atualização do grafo.

Antes de iniciar a execução completa do algoritmo de criação ou atualização do grafo, é necessário que para cada base de dados, no caso as bases B1 e B2 previamente apresentadas, sejam desenvolvidos os métodos para pré-processamento dos dados. Esses métodos devem possuir uma assinatura padrão para que sejam executados de maneira automática sempre que for solicitada a atualização do grafo de classificação.

O algoritmo de criação ou atualização do grafo começa executando os métodos de pré-processamento dos dados que retornam a lista de sensores e os estados finais, bem como os dados brutos tratados em uma estrutura de matriz. Em seguida é executado o método de

Figura 30 – Solicitação de criação ou atualização do grafo de classificação



Fonte: Próprio autor

extração de *features* que retorna o nome das *features* para cada tipo de sensor, seguindo o padrão "<sensor>_<nome da *feature*>" (por exemplo, as *features* referentes a média para o acelerômetro, são identificadas como acc_mean). Além dos nomes, também são retornados os dados processados das *features* (as *features* extraídas). O conjunto de dados de *features* são divididos em dois subconjuntos: um subconjunto de treinamento e um subconjunto de teste (foi usada a proporção de 90% e 10%, respectivamente). As *features* e os estados finais são então usados como entradas para o treinamento dos algoritmos de classificação. Em seguida, os modelos inteligentes treinados são avaliados (com o conjunto de teste) um a um, e os nomes dos modelos treinados, bem como a probabilidade de atingir cada um dos estados finais usando cada um dos modelos treinados são retornados. Para a PoC, foi utilizada a probabilidade de acurácia.

Finalmente, os tipos de sensores, nomes das *features*, nomes de modelos inteligentes treinados e os estados finais e seus relacionamentos que compõem o grafo de classificação são armazenados no banco de dados. Além disso, também é gerado um arquivo ".xml" com essa versão completa do grafo de classificação criado. Esse arquivo ".xml" é armazenado em um diretório específico no servidor e pode ser baixado via requisição a API. Além do grafo, os modelos treinados também são armazenados em um diretório no servidor em arquivos ".tflite" (para os modelos gerados pelo algoritmo de rede neural artificial) e ".onnx" (para os modelos gerados pelos algoritmos árvore de decisão e *random forest*). Também é possível executar o *download* desses modelos através de uma requisição a API.

5.1.2.5 Algoritmos de Classificação

Usando as *features* extraídas como entrada, foram executados três algoritmos de classificação: uma rede neural artificial com quatro camadas, uma árvore de decisão e uma *random forest*. Para a primeira base de dados (B1), foram treinados três modelos (um para cada algoritmo de classificação) usando as *features* extraídas de dados de acelerômetro e giroscópio. Nesse caso, os modelos treinados são usados para inferência dos 17 estados finais da base de dados B1. Também foram treinados outros três modelos (novamente um para cada algoritmo de classificação) usando como entrada as *features* extraídas apenas dos dados do acelerômetro coletados das duas bases de dados. Neste segundo caso, os modelos treinados são usados para inferência dos 30 estados finais coletados das duas bases de dados, os 17 estados finais da base B1 e os 14 estados finais da base de dados B2. Note que foram obtidos 30 estados finais e não 31, pois há um tipo de movimento (andar) que está presente em ambas as bases de dados.

Os modelos treinados foram implementados usando as bibliotecas TensorFlow (DAVID *et al.*, 2021) para o algoritmo rede neural artificial, e Scikit-Learn (PEDREGOSA *et al.*, 2011) para os algoritmos árvore de decisão e *random forest*. Essas bibliotecas foram escolhidas, pois ambas também oferecem versões simplificadas (Tensorflow lite (DAVID *et al.*, 2021) para modelos TensorFlow e Onnx (ASIF, 2021) para modelos Scikit-Learn) com baixos custos de processamento e de energia para serem executadas em aplicações Android e IOS, desenvolvidas em linguagens JAVA, Kotlin ou Swift.

Isso permite que, uma vez baixados os modelos treinados, eles podem ser reutilizados para inferência dos movimentos e situações de saúde, usando *features* extraídas de dados coletados de sensores dos dispositivos móveis em tempo real. Vale destacar que os modelos em formato ".onnx" também podem ser convertidos para modelos em formato ".tflite", caso os desenvolvedores queiram utilizar apenas uma biblioteca em suas aplicações.

Na Tabela 4 são apresentados os resultados médios da probabilidade de acurácia obtida para os algoritmos de classificação utilizados nos primeiros testes. O modelo inteligente treinado usando o algoritmo de *random forest* obteve melhor resultado nos dois cenários verificados, enquanto que no cenário 1, onde foram utilizados os sensores combinados de acelerômetro e giroscópio, o algoritmo de rede neural artificial apresentou resultado similar ao algoritmo de árvore de decisão e, no cenário 2, onde foi utilizado apenas o sensor de acelerômetro, o algoritmo de rede neural se saiu melhor do que o de árvore de decisão. Esses resultados podem estar diretamente relacionados à natureza dos dados e ao número de instâncias das duas bases de

dados utilizadas para construção do grafo de classificação, que chegam a algumas centenas de medições em ambas as bases. Vale ressaltar que o algoritmo de *random forest* também foi o que apresentou melhores resultados no trabalho de (LINHARES *et al.*, 2020), que foi a fonte onde foi obtida a base de dados B1.

Tabela 4 – Algoritmos de classificação implementados

Algoritmo	**Sensores**	**N. Estados finais**	**Acurácia**
Rede Neural Artificial	Acelerômetro, Giroscópio	17	0.75
Árvore de Decisão	Acelerômetro, Giroscópio	17	0.75
Random Forest	Acelerômetro, Giroscópio	17	0.85
Rede Neural Artificial	Acelerômetro	30	0.6
Árvore de Decisão	Acelerômetro	30	0.55
Random Forest	Acelerômetro	30	0.65

Fonte: Próprio Autor

Em (LINHARES *et al.*, 2020), onde foi utilizada a primeira base de dados, obteve-se uma melhor probabilidade de acurácia para esses mesmos algoritmos de classificação exclusivamente para detecção de quedas. No entanto, acredita-se que essa diferença ocorreu por utilizarmos os dados para inferir todos os tipos de movimentos presentes na base de dados (modelo de inferência multi-classe), enquanto que no trabalho de (LINHARES *et al.*, 2020) buscou-se gerar a detecção de apenas um movimento, a queda, de modo que todos os demais dados foram tratados como parte de um conjunto de movimentos de não queda (modelo de inferência binário). Além disso, em (LINHARES *et al.*, 2020), o pré-processamento dos dados é seguido por uma etapa de enriquecimento desses dados. Para essa PoC foi desenvolvido um algoritmo de pré-processamento mais simplificado, já que o foco era avaliar a viabilidade da construção e uso do grafo de classificação e não a acurácia dos algoritmos utilizados. No entanto, a análise do trabalho de (LINHARES *et al.*, 2020) permite supor que é possível obter melhores valores de probabilidade para o grafo de classificação com os mesmos algoritmos de classificação, realizando ajustes nas etapas de pré-processamento dos dados.

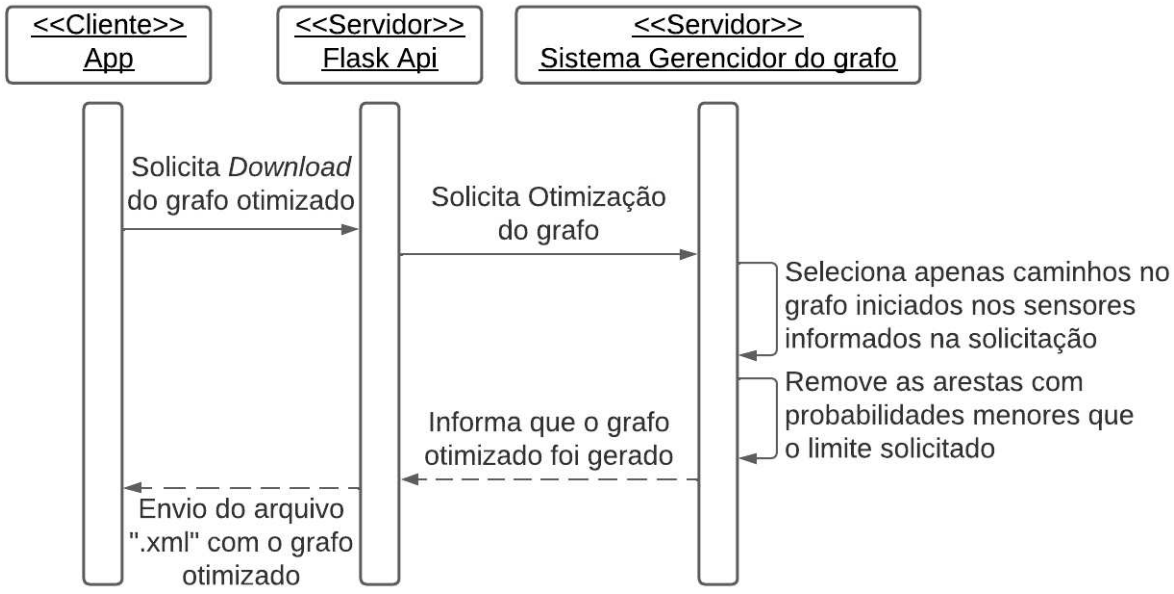
5.1.2.6 API Web e Otimização do Grafo

A API Web foi implementada para que as aplicações pudessem requisitar o download do grafo de classificação e dos modelos treinados, ou para requisitar a atualização do grafo de classificação. Essa API recebe solicitações externas e aciona o sistema de gerenciamento do grafo de classificação implementado para atualizar, otimizar e executar o *download* do grafo de

classificação e dos modelos inteligentes treinados. No caso da solicitação de atualização de grafo, a API retorna a informação que o grafo deve ser atualizado para o sistema de gerenciamento do grafo. Esse sistema verifica então se há uma solicitação de atualização do grafo fornecida pela API a cada minuto. Em seguida, o sistema de gerenciamento do grafo executa os algoritmos de pré-processamento, treinamento dos algoritmos de classificação e geração e armazenamento no banco de dados do grafo de classificação.

No caso de uma aplicação requisitar à API o *download* do grafo de classificação otimizado, a API solicita ao sistema de gerenciamento do grafo que uma versão otimizada do grafo seja montado e retorna para aplicação o arquivo ".xml" como o grafo de classificação otimizado. Neste caso específico, a aplicação que está requisitando o *download* do grafo de classificação otimizado, passa como parâmetros para API a lista de sensores utilizados pela aplicação e, opcionalmente, um *threshold* mínimo para a probabilidade de atingir as possíveis situações de saúde (ou estados finais) com base nos modelos treinados usando os algoritmos de classificação. A Figura 31 ilustra as etapas de otimização e download do grafo de classificação otimizado.

Figura 31 – Solicitação de download do grafo de classificação otimizado



Fonte: Próprio autor

A otimização do grafo de classificação segue o processo apresentado na Seção 4.2.1.4. Primeiro, é gerada uma versão parcial em memória do grafo otimizado usando os dados do grafo armazenado no banco de dados MYSQL, contendo apenas as entidades referentes aos sensores

e arestas que contemplam os caminhos que iniciam com os tipos de sensores informados na requisição feita à API. Essa versão parcial do grafo contém os sensores, as *features* geradas com base nos dados dos tipos de sensores informados, os modelos dos algoritmos de classificação treinados que utilizaram as *features* selecionadas como entrada e os estados finais alcançados por esses modelos. Em seguida, são removidas as arestas que ligam os modelos treinados e os estados finais que contêm um valor de probabilidade menor que o *threshold* informado. Por fim, é gerado o arquivo ".xml" contendo o grafo otimizado e o link para *download* do mesmo é enviado à API, que por sua vez envia o link a aplicação que que fez a solicitação.

É possível também solicitar a API o grafo de classificação completo, nesse caso, a API aciona o sistema de gerenciamento do grafo. O sistema de gerenciamento, nesse caso, apenas envia à API o link para *download* do grafo de classificação completo, que já está armazenado em um diretório no servidor. Finalmente, a API retorna esse mesmo link à aplicação que fez a solicitação.

Por último, também é possível requisitar à API o *download* dos modelos inteligentes treinados quando o grafo de classificação foi criado ou atualizado pela última vez (no caso dessa PoC, os modelos de rede neural artificial, árvore de decisão e *random forest*). Para fazer o download, a aplicação solicitante informa à API o nome do modelo que deseja fazer o *download* (os nomes dos modelos estão contidos no grafo de classificação). Por sua vez, a API retorna o link para *download* do modelo inteligente armazenado em um diretório no servidor.

5.1.2.7 Exemplo de uso do Grafo de Classificação

Para testar o uso do grafo de classificação no auxílio do desenvolvimento de uma aplicação de IoHT e como base de conhecimento, foi utilizada a instância do grafo de classificação que foi implementada para evoluir a aplicação android dessa primeira PoC. Para isso, primeiro foi feito o *download* da versão otimizada do grafo de classificação e foi implementado um método usando a biblioteca TensorFlow Lite (DAVID *et al.*, 2021) para inferência do movimento, baseada no modelo inteligente que foi treinado usando a rede neural artificial. Com isso, a aplicação se torna capaz de monitorar e inferir automaticamente os movimentos executados pelo usuário. Posteriormente, a aplicação foi modificada para que fosse feito o *download* automático do arquivo contendo o grafo de classificação otimizado para um diretório no dispositivo móvel.

Como a aplicação utiliza apenas o sensor de acelerômetro, o grafo de classificação otimizado usado contém apenas os caminhos que iniciam nos vértices correspondente ao sensor

do acelerômetro. Após o *download* do grafo de classificação, a aplicação preenche um objeto de representação do conhecimento com base nesse grafo. Então, a lista de modelos treinados é verificada e também é solicitado o *download* da API para cada um, embora, como dito anteriormente, para simplificação só foi utilizado para inferência dos movimentos o modelo inteligente treinado com a rede neural artificial. Os arquivos contendo o grafo de classificação e os modelos treinados baixados possuem em conjunto em torno de 10 Mb. Em média, nos testes, o download automático do grafo e dos modelos demorou entre 0,5 e 2 segundos usando uma conexão de internet banda larga de 300 Mb/s e as predições dos movimentos, usando o dados coletados em tempo real pelo sensor de acelerômetro do *smartphone* usado para teste e o modelo inteligente, em média foram executadas em menos de 1s.

Com esta aplicação, foi possível avaliar a viabilidade do uso do grafo de classificação como base de conhecimento e assim auxiliar na implementação da aplicação. Ainda assim, não foi possível avaliar o impacto na velocidade de desenvolvimento de uma aplicação IoHT quando é utilizado o grafo de classificação.

5.1.3 Resultados da primeira PoC

Com a aplicação Android desenvolvida nessa PoC foi possível identificar diferentes atividades/padrões de movimentação usando um sensor de acelerômetro e apresentar alertas com recursos que se adaptam de acordo com o contexto de bateria do dispositivo. Desse modo, essa PoC permitiu verificar a viabilidade do uso da versão preliminar do processo MOTION para auxiliar o desenvolvimento de sistemas IoHT autoadaptativos baseados em padrões de movimento. Além disso, também foi possível verificar a viabilidade e funcionalidades do uso de uma instância do GRAFIT.

Como resultado da PoC foi possível também reavaliar e refinar o processo MOTION e prover uma base para construção do *framework* KREATION, bem como analisar artefatos de reúso para aproveitamento dentro desse *framework*.

Após o uso da versão preliminar do processo para guiar o desenvolvimento da aplicação Android desenvolvida nessa PoC, o processo MOTION foi refinada para a versão ilustrada na Figura 15, que foi descrita no capítulo 4 desse documento. As principais mudanças foram: (i) a fusão das atividades referentes a escolha dos estados de saúde que a aplicação deve monitorar e escolha dos sensores que serão utilizados pela aplicação; (ii) a separação da atividade única de Verificação e Validação nas atividades de Verificação e Validação em tempo

de projeto e Validação e Verificação em tempo de execução, tornando mais claro que a validação e verificação da aplicação deve ser feita em dois momentos distintos; (iii) a adição dos *loops*, de modo a demonstrar visualmente que o processo MOTION pode ser utilizado em conjunto com modelos de processo de desenvolvimento de *softwares* iterativos. Além disso, o manual de uso do processo MOTION foi atualizado para incorporar as mudanças mencionadas anteriormente.

Com a aplicação Android desenvolvida nessa PoC foi possível identificar diversos elementos que poderiam ser replicados e utilizados na construção do *framework* KREATION, como os módulos referentes ao *loop* de adaptação MAPE-K e o *framework* SUCCEEd. Vale ressaltar que por ser desenvolvido na linguagem Java, o *framework* SUCCEEd pode ser facilmente incorporado ao *framework* KREATION, desenvolvido em Kotlin, uma vez que a linguagem Kotlin é baseada na linguagem Java e por isso é possível utilizar códigos fonte em Java juntamente com o código em Kotlin. No entanto, com a aplicação Android desenvolvida, percebeu-se que não seria possível reutilizar por completo o *midleware* LoCCAM em sua versão atual no *framework* KREATION, pois ela está restrita a versões antigas do sistema operacional Android e possui certas exigências de acesso a memória externa do dispositivo que são impeditivas para alguns dispositivos Android mais recentes. Porém, parte da arquitetura do *midleware* LoCCAM foi utilizada como base para construção do módulo referente a etapa de monitoramento do *loop* MAPE-K do *framework* KREATION.

Ao desenvolver a aplicação Android dessa PoC foi também identificado que seria vantajoso o uso do padrão de *software observer* para comunicação entre as etapas do *loop* de adaptação. Além disso, percebeu-se ao utilizar o GRAFIT, que seria interessante que o *framework* KREATION possuísse métodos para facilitar o uso do mesmo em conjunto com os demais artefatos propostos nessa tese.

5.2 Segunda prova de conceito

Nessa segunda PoC, buscou-se avaliar a viabilidade e funcionalidade do *framework* KREATION em conjunto com o *template* ARTe e o grafo de classificação. Pata tal, foram desenvolvidas duas aplicações Android utilizando o processo MOTION e implementadas ambas usando como base o *framework* KREATION. Para cada aplicação, diferentes regras de adaptação foram construídas usando o *template* ARTe. Além disso, foi reutilizado a instância do grafo de classificação gerada na primeira PoC.

A primeira aplicação, intitulada *Risk movEment ALERt* (REALER), identifica pa-

drões de movimento com base nos dados de acelerômetro de um *smartphone*. A segunda aplicação, chamada *Activity Recognition and Cardiology ANALlysis* (ARCANA), identifica padrões de movimento com base em dados de acelerômetro e giroscópio de um *smartphone* e monitora a frequência cardíaca por meio de dados de *smartwatch* enviados ao Google Fit. Ambas as aplicações foram implementadas utilizando a IDE Android Studio em um notebook com processador AMD Ryzen 7 5700U 1.8Ghz e 8 GB de memória RAM.

O grafo de classificação instanciado na primeira PoC foi utilizado para criar um objeto de representação do conhecimento (`KnowledgeRepresentation`) para ambas as aplicações. Vale destacar que o *framework* KREATION possui métodos para fazer o *download* automático da instância do grafo de classificação e dos respectivos modelos inteligentes treinados para o dispositivo onde a aplicação está executando, através de requisições a uma API Web que se conecta ao sistema de gerenciamento do grafo de classificação no servidor.

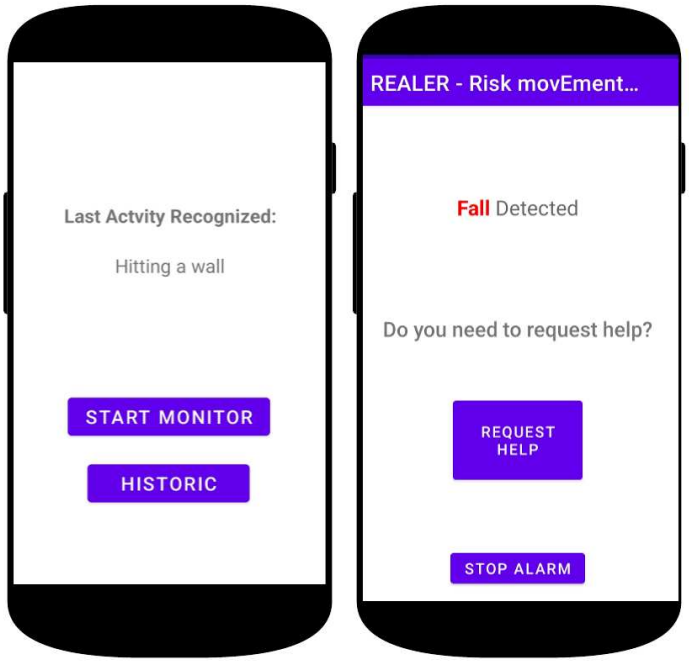
5.2.1 Aplicação REALER

A aplicação REAELER tem como objetivo alertar situações de risco (quedas ou esbarrões) e registrar padrões de movimento com base nos dados do acelerômetro. Caso seja identificada uma situação de risco, a aplicação dispara um alerta utilizando os recursos do dispositivo móvel (texto, vibração e luz do *flash* da câmera). O usuário do aplicativo pode, nesse caso, solicitar ajuda de um cuidador cadastrado ou parar o alarme. E, caso o usuário não realize nenhuma ação na tela de alarme, os cuidadores cadastrados são automaticamente informados após um período. Os recursos disponíveis para o alerta são adaptados automaticamente de acordo com o nível de bateria do dispositivo móvel. Caso a bateria esteja abaixo ou acima de determinados níveis especificados nas regras de adaptação, um subconjunto de recursos diferente é usado. Além disso, a inferência de qual movimento foi realizado é baseada nos modelos inteligentes treinados informados na instância do grafo de classificação. O código fonte da aplicação REALER está disponível em: https://bit.ly/3N6viCW.

A funcionalidade de alerta do *app* REALER pode ser adaptada automaticamente como citado previamente. Essa adaptação consiste na modificação dos recursos utilizados para o alerta sempre que o nível de bateria atinge ou se torna menor que certos *thresholds*. Caso o nível restante da bateria seja superior a 70%, o aplicativo usará luzes do *flash* da câmera do *smartphone*, vibração e texto no alerta. Se o nível de bateria restante for maior que 40% e menor que 70%, o aplicativo usará apenas a vibração e o texto para o alerta. E, se o nível de bateria

restante for inferior a 40%, o aplicativo usará apenas textos no alerta.

Figura 32 – Exemplos de telas da aplicação REALER



Fonte: Próprio autor

As regras de adaptação do *app* REALER estão definidas em um arquivo gerado usando o *template* ARTe. Essas regras podem ser alteradas diretamente nesse arquivo sem que haja necessidade de modificar a aplicação, de modo que essa modificação pode ser feita inclusive em tempo de execução. Para isso foi criada um API Web contendo o arquivo de regras de adaptação em um servidor de nuvem localizado em uma instância EC2 da AWS. Desse modo, a aplicação pode fazer uma requisição a essa API para executar o *download* automático do arquivo de regras de adaptação de tempos em tempos. O *framework* KREATION também possui métodos para *download* e manipulação dos arquivos de regras de adaptação construídos utilizando a estrutura do *template* ARTe.

A REALER é uma aplicação feita com base em requisitos similares aos da aplicação Android da primeira PoC, desse modo, ela funciona como uma reconstrução da aplicação anterior utilizando o *framework* KREATION, como proposto na fase final do método de construção de *frameworks* baseados em Análise de Domínio.

A Figura 32 apresenta exemplos de duas telas do aplicativo REALER. Na esquerda está a tela principal da aplicação, onde é possível iniciar o monitoramento ou acessar a tela de histórico com as atividades registradas e situações de risco detectadas. Apenas as atividades previamente definidas são registradas. Além disso, a última atividade detectada é mostrada

na tela. Caso seja identificada uma situação de risco, a tela da direita é chamada e o alerta é iniciado utilizando os recursos disponíveis de acordo com o nível de bateria do aparelho. O alerta é interrompido quando o botão <*Stop Alert*> é ativado e nesse caso o usuário retorna à tela principal. Também é possível nesta tela de alerta informar a um cuidador cadastrado sobre a situação de risco detectada através do botão <*REQUEST HELP*>. Na tela principal, quando a atividade identificada não corresponde a uma situação de risco, o monitoramento continua em execução durante o uso do *app*.

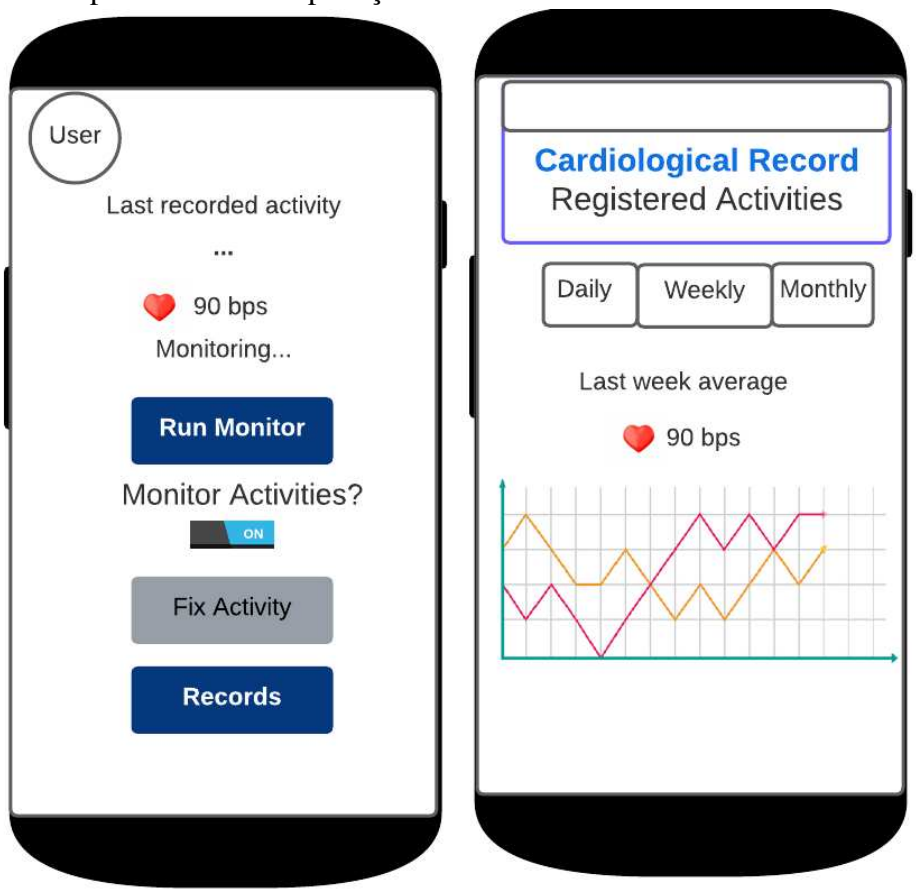
5.2.2 Aplicação ARCANA

A ARCANA é uma aplicação de reconhecimento de atividades do dia a dia e análise da frequência cardíaca. O reconhecimento das atividades é baseado nos sensores acelerômetro e giroscópio presentes no dispositivo móvel. Já a análise da frequência cardíaca é coletada por um *smartwatch* que envia os dados para o aplicativo de gerência dos dados do dispositivo (que é definido pela fabricante do *smartwatch*) no *smartphone* Android. O aplicativo de gerenciamento se conecta com a API do Google Fit e envia os dados coletados pelo *smartwatch*. A ARCANA obtém então os dados através de uma requisição a API do Google Fit. O código fonte da aplicação ARCANA está disponível em: https://bit.ly/42INfxk.

Quando o aplicativo ARCANA identifica que o nível da frequência cardíaca está fora do normal por um período, o paciente é notificado e pode solicitar ajuda imediata para emergências ou cuidadores cadastrados. O aplicativo também envia automaticamente o alerta para os cuidadores cadastrados e relatórios de atividades reconhecidas próximas ao período em que foi identificada a frequência cardíaca anormal.

O alerta mostrado ao usuário também possui diferentes recursos (luz, vibração, áudio e texto) que são ativados e desativados, adaptando-se de acordo com as mudanças no nível da bateria. A aplicação ARCANA também utiliza um arquivo gerado com base no *template* ARTe, para definição de suas regras de adaptação. Esse arquivo também é disponibilizado pela mesma API Web utilizada pelo *app* REALER. Para a aplicação ARCANA, quando o nível restante da bateria for superior a 70%, a funcionalidade de alerta utilizará luzes do *flash* da câmera do *smartphone*, som, vibração e texto. Se o nível de bateria restante for maior que 50% e menor que 70%, o aplicativo usará som, vibração e o texto para o alerta. Se o nível de bateria for maior que 40% e menor que 50%, o aplicativo usará apenas a vibração e o texto para o alerta. E, se o nível de bateria restante for inferior a 40%, o aplicativo usará apenas texto para o alerta.

Figura 33 – Exemplos de telas da aplicação ARCANA



Fonte: Próprio autor

A inferência da atividade realizada no aplicativo ARCANA também é baseada nos modelos inteligentes treinados da instância do grafo de classificação. A análise da frequência cardíaca utiliza um algoritmo simples que verifica se o valor da frequência cardíaca está acima ou abaixo de um *threshold* considerado normal. Além disso, caso o *app* reconheça erroneamente uma atividade, o usuário pode informar a atividade correta. O aplicativo armazena *features*, como média, máximo, mínimo e desvio padrão, gerados a partir dos dados coletados pelos sensores usados para inferir a atividade. Esses dados são usados para treinar um algoritmo de inferência interno do aplicativo com base em limites. O usuário pode escolher se essa atividade específica usará o algoritmo de limite para sua inferência ou se a inferência será feita com base nos modelos inteligentes do grafo de classificação.

A Figura 33 apresenta exemplos dos protótipos de telas do aplicativo ARCANA. Na esquerda está a tela principal do *app*, onde você pode começar a monitorar seus batimentos cardíacos e atividades. Quando uma atividade é reconhecida, o botão <*Fix Activity*> é habilitado para que, se necessário, o usuário possa corrigir a atividade relatada. Também é possível acessar o menu de navegação clicando no ícone do usuário ou acessar a tela de histórico de *logs* mostrada

na tela à direita. Na tela à direita, o usuário pode ver o histórico de frequência cardíaca e atividades nos períodos diário, semanal e mensal.

5.2.3 Desenvolvimento das aplicações da segunda PoC

Seguindo o processo MOTION, para ambas as aplicações, primeiro foram definidos os estados de saúde que se deseja monitorar e os sensores que seriam utilizados. No caso do *app* REALER foi definido que seriam utilizados apenas sensores presentes no *smartphone*, enquanto que para o *app* ARCANA foram utilizados os sensores do *smartphone* e de um *smartwatch*. Em seguida, foram elicitados os demais requisitos, as adaptações de cada aplicação foram definidas e as regras de adaptações foram geradas e salvas em um arquivo seguindo o *template* ARTe.

Sugere-se que, após a definição dos requisitos, seja gerado uma instância do modelo de grafo de classificação GRAFIT a ser utilizado pela aplicação. No caso das duas aplicações REALER e ARCANA, foi reutilizado o mesmo grafo de classificação que foi gerado para a primeira PoC. Em seguida, foram criados projetos simples de cada uma das aplicações móveis, contendo também os protótipos de telas e as transições entre cada uma dessas telas.

Com base nos projetos gerados, as aplicações foram então implementadas utilizando o *framework* KREATION. Para isso, foi feito o *download* do código do *framework*. Em seguida, os arquivos de configuração do aplicativo base do *framework* foram modificadas para cada uma das aplicações dessa PoC. Com o uso do *framework* KREATION foi possível reaproveitar toda a lógica do ciclo de adaptação, inclusive a inferência automática, utilizando o arquivo contendo as regras de adaptação e o grafo de classificação. Para baixar e utilizar o grafo de classificação, os modelos inteligentes e o arquivo de regras de adaptação, foi necessário alterar o valor das constantes do *framework* que contemplam os links de acesso às APIs utilizadas para solicitar o *download* desses artefatos de *software*.

Para ambas as aplicações foi utilizada a API de *login* do Google, reaproveitando os códigos desta funcionalidade presentes no *framework* KREATION. E para aplicação ARCANA foram reutilizados os métodos contidos no *framework* KREATION para acesso da API Google FIT com o intuito de obter os dados coletados pelo *smartwatch*.

Após a implementação, foram feitos testes das funcionalidades principais das aplicações, incluindo a autoadaptação com base no contexto de bateria, em dois *smartphones*, um Xiaomi Redmi 10 e um Sansung Galaxy M53.

Com o auxilio do *framework* KREATION a implementação do *back-end* das aplica-

ções foi facilitada, de modo que o principal esforço na implementação do *back-end* se concentrou na lógica de alertas específica de cada *app* e das funcionalidades dos elementos visuais em cada tela, como botões e menus. Assim sendo, ao final do desenvolvimento, as atividades de implementação mais onerosas foram a criação das telas e a implementação do *front-end* das aplicação, enquanto que a implementação do *back-end* das duas aplicações exigiu pouco esforço em comparação ao *front-end*.

5.2.4 Resultados da segunda PoC

Com o resultado do desenvolvimento das aplicações dessa PoC foi possível demonstrar a viabilidade da versão atualizada do processo MOTION após a primeira PoC, para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento, bem como a viabilidade e funcionalidade do *framework* KREATION para auxiliar implementação de aplicações IoHT autoadaptativas.

Após finalizada a segunda PoC não foi identificada a necessidade de nenhuma modificação maior no processo MOTION, apenas foi feita uma atualização no texto do Manual de Execução do processo para a atividade de implementação.

Finalmente, com os resultados da PoC foram feitas algumas atualizações no manual de uso do *framework* KREATION para facilitar o reúso desse artefato.

5.3 Conclusão

Este capítulo mostrou como o processo MOTION e os artefatos de reúso (modelo GRAFIT, *template* ARTe e *framework* KREATION) foram avaliados em relação a sua viabilidade e funcionalidade através do desenvolvimento de provas de conceito.

A primeira versão do processo MOTION foi avaliada utilizando a primeira prova de conceito apresentada neste capítulo. Essa avaliação permitiu identificar diversos pontos de melhoria, provendo assim o processo refinado que foi apresentado no capítulo 4 dessa tese.

No próximo capítulo é apresentado um experimento feito com profissionais de TIC que permitiu avaliar, com base na percepção dos participantes do experimento, a utilidade e facilidade de uso do processo MOTION.

6 EXPERIMENTO

Neste capítulo é apresentado um experimento com profissionais de TIC, que tem como base a metodologia de avaliação de processos de propósito geral proposta por (SHULL *et al.*, 2001) e a estratégia para experimentação em engenharia de *software* de (WHOLIN *et al.*, 2000; WOHLIN *et al.*, 2012). O intuito desse experimento é avaliar a utilidade e a facilidade de uso do processo MOTION para auxiliar o desenvolvimento de aplicações autoadaptativas de IoHT baseadas em padrões de movimento pela perspectiva de profissionais de Tecnologia da Informação e Comunicação (TIC). Os *feedbacks* dos profissionais de TIC e os *logs* coletados durante o experimento foram utilizados para a análise do processo MOTION. A análise dos dados foi feita com base em medidas utilizadas em outros estudos da literatura que buscam analisar tanto processos de desenvolvimento já consolidados quanto a aceitação de novas tecnologias.

Na Seção 6.1 é apresentado o planejamento do experimento. A execução do experimento é detalhada na Seção 6.2. Na Seção 6.3 são apresentados os resultados do experimento, os quais são discutidos na Seção 6.4. As Ameaças à Validade do experimento são discutidas na Seção 6.5. Por fim, na Seção 6.6 é apresentada a conclusão do capítulo.

6.1 Planejamento do Experimento

Para escolher o instrumento de avaliação do processo MOTION, primeiro foi feita uma pesquisa na literatura sobre avaliações de processos de desenvolvimento de *software*, na qual foram identificados alguns estudos com avaliações para processos de desenvolvimento de *software* já consolidados, como processos cascata ou processos iterativos usando a metodologia SCRUM (DIXIT; BHUSHAN, 2019; CUNHA *et al.*, 2006), porém, não foi identificado nenhuma avaliação voltada para novos processos de desenvolvimento de *software*. Foram então analisadas as estratégias empregadas nesses estudos, bem como a estratégia de avaliação de processos em geral proposta em (SHULL *et al.*, 2001), que, a princípio, não é voltada a processos de desenvolvimento de *software*. Além disso, também foram analisadas estratégias de aceitação de novas tecnologias, como o *Technology Acceptance Model* (TAM) (MARIKYAN; PAPAGIANNIDIS, 2022).

Com base nesses estudos foi elaborado o planejamento do experimento feito nessa pesquisa. Esse planejamento foi submetido a análise de seis especialistas de domínio com experiência em avaliação de sistemas de *software* diversos, incluindo três especialistas em

sistemas IoHT.

A avaliação do planejamento do experimento foi feita em três etapas. Primeiro, o planejamento inicial do experimento bem como a descrição do processo MOTION e o Manual de Execução do processo MOTION foram enviados aos especialistas. Segundo, os *feedbacks* iniciais fornecidos pelos especialistas foram incorporados ao planejamento. Por último, foram feitas reuniões com os especialistas para que, em comum acordo, fosse obtida a versão final do planejamento do experimento.

Nas próximas subseções são apresentados os objetivos, as hipóteses, os pré-requisitos e as fases do experimento planejado.

6.1.1 Objetivos e Questões Avaliadas no Experimento

Um dos objetivos deste experimento é verificar se o uso do processo de desenvolvimento MOTION impacta no desempenho do time durante o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Com base em (SHULL *et al.*, 2001), por impacto no desempenho do time, entende-se que o uso do processo pode auxiliar e/ou agilizar o desenvolvimento desse tipo de aplicação, bem como pode auxiliar o gerenciamento de atividades.

Além disso, espera-se que o processo MOTION seja útil e fácil de usar na perspectiva dos profissionais de TIC, de modo que possa ser adotado tanto por profissionais com pouco conhecimento no desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento, quanto por profissionais mais experientes no desenvolvimento desse tipo de aplicação.

Considerando os objetivos da avaliação, foram formuladas duas questões que guiaram o experimento executado.

A primeira questão visa avaliar o impacto do uso do processo MOTION no desempenho do time durante o desenvolvimento da aplicação IoHT autoadaptativa baseada em padrões de movimento. Essa questão foi formulada da seguinte maneira:

Questão 1: O processo MOTION é capaz de impactar o desempenho do time de desenvolvimento e auxilia a gerência das atividades do processo de desenvolvimento de *softwares* IoHT autoadaptativos baseados em padrões de movimento?

Considerando esta questão foi gerado o seguinte conjunto de hipóteses:

– **Hipótese Nula 1**: Usar o processo MOTION não impacta ou impacta negativamente

no desempenho, produtividade, estabilidade de requisitos e satisfação do time durante o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

– **Hipótese Alternativa 1**: O processo MOTION impacta positivamente no desempenho, produtividade, estabilidade de requisitos e satisfação do time durante o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

Esse conjunto de Hipóteses permite avaliar se há impacto positivo do uso do processo MOTION em termos de desempenho no desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

A segunda questão formulada visa avaliar o processo MOTION sob os aspectos da utilidade e facilidade de uso na visão dos profissionais de TIC. Sendo a questão a seguinte:

Questão 2: Na perspectiva dos profissionais de TIC, o processo MOTION é útil e fácil de ser usado?

Considerando o objetivo do processo MOTION, essa questão visa identificar se os profissionais de TIC participantes do experimento percebem a utilidade do processo MOTION e se entendem que ele atende ao que se propõe. Para analisar essa questão foi elaborado um questionário baseado no modelo TAM (SILVA, 2015; MARIKYAN; PAPAGIANNIDIS, 2022) com respostas na escala Likert (JOSHI *et al.*, 2015).

Com base nessa questão foram elaboradas dois conjuntos de hipóteses para avaliar a utilidade e facilidade de uso com base nos dados médios coletados pelo questionário. Esses conjuntos de hipóteses são:

– **Hipótese Nula 2**: Na perspectiva dos profissionais de TIC que utilizaram o processo MOTION, ele não é útil ou não é possível identificar a utilidade do processo MOTION para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

– **Hipótese Alternativa 2**: Na perspectiva dos profissionais de TIC que utilizaram o processo MOTION, ele é útil para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

– **Hipótese Nula 3**: Na perspectiva dos profissionais de TIC que utilizaram o processo MOTION, ele não é fácil de usar ou não é possível identificar a facilidade de uso do processo MOTION para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

– **Hipótese Alternativa 3**: Na perspectiva dos profissionais de TIC que utilizaram o processo MOTION, ele é fácil de usar para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

Para analisar as questões e hipóteses apresentadas anteriormente, foram escolhidas medidas quantitativas e qualitativas a serem coletadas durante o experimento, possibilitando uma análise estatística que permitisse avaliar se é possível rejeitar ou não as hipóteses nulas em detrimento das hipóteses alternativas. Essas medidas são descritas na subseção 6.1.4.2.

6.1.2 Materiais

O desenvolvimento das aplicações foi feito com a utilização dos *notebooks* dos próprios profissionais de TIC convidados, permitindo que o experimento fosse feito inteiramente de maneira remota. Os integrantes dos times também optaram por utilizar seus próprios aparelhos Android ou simuladores para os testes.

Também foram fornecidos, a cada time, pelo pesquisador e autor da tese, dois dispositivos *smartwatch Amazfit Bip U*, com Amazfit OS, capacidade de armazenamento da memória interna de 2300 MB, conexão com dispositivos Android (versão 5 ou superior) e IOS (versão 10 ou superior). Esse *smartwatch* possui monitoramento de atividades e sensores de monitoramento de frequência cardíaca, sono, GPS e *bluetooth*.

Além disso, os artefatos de reúso de *software* desenvolvidos na pesquisa também foram disponibilizados para os participantes do experimento, bem como suas descrições, manuais de uso e exemplos de códigos de aplicações desenvolvidas usando esses artefatos. Toda a documentação utilizada pelos profissionais de TIC durante o experimento, bem como os vídeos de treinamento e link para códigos e manuais dos artefatos de *software* e exemplos de uso está disponível em https://bit.ly/48QXT8X.

Por fim, também foi disponibilizado a descrição e o manual de execução do processo MOTION, o qual pode ser visto no Apêndice A.

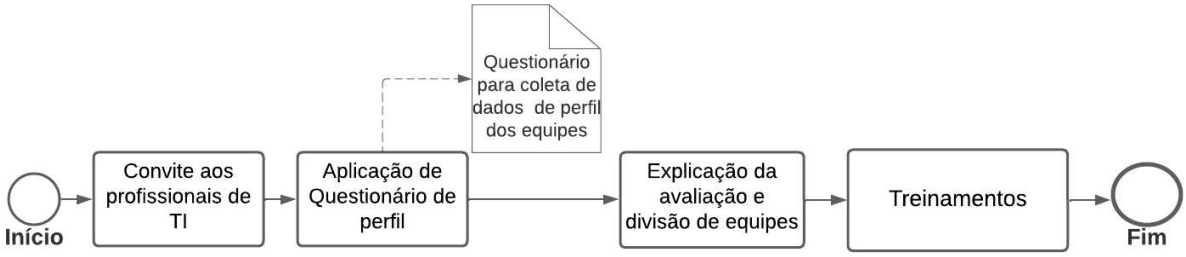
6.1.3 Planejamento das fases do experimento

Esse experimento foi dividido em duas fases: inicialmente, os participantes passaram por um treinamento para uso do Processo MOTION e dos artefatos de reúso, e depois ocorreu a fase de execução e coleta de dados. Nas subseções seguintes são descritas as atividades de cada um dessas fases.

6.1.3.1 Planejamento da fase de preparação

A Figura 34 ilustra a fase de preparação do experimento.

Figura 34 – Fase de preparação do experimento



Fonte: Próprio autor

Primeiro foi planejada a participação de no mínimo seis e no máximo dez profissionais de TIC no experimento. Os participantes foram divididos em dois times com números iguais de membros. Para cada um dos profissionais convidados foi considerado o seguinte perfil mínimo:

- Possuir experiência de pelo menos 1 ano com o desenvolvimento de aplicações para dispositivos móveis em Android;
- Possuir experiência com a utilização de processos de desenvolvimento de *software*;
- Possuir experiência com trabalho em equipes de desenvolvimento; e
- Ter disponibilidade de pelo menos 4h semanais durante o período de seis semanas (período da execução do experimento).

Embora o processo MOTION não tenha necessariamente foco em aplicações Android, a necessidade de que os profissionais de TIC convidados tivessem experiência com Android está relacionada as aplicações a serem desenvolvidas durante o experimento. Além disso, o *framework* KREATION tem como foco o desenvolvimento de aplicações Android e ele é utilizado para facilitar o desenvolvimento das aplicações durante o experimento. Também era desejável que parte dos profissionais de TIC convidados tivessem conhecimento de utilização de sensores e atuadores IoT e já tivessem desenvolvido pelo menos uma aplicação de IoHT. Também era esperado que os profissionais de TIC convidados conhecessem processos de desenvolvimento de *software* e metodologias ágeis, pois, com isso, a compreensão do processo de desenvolvimento MOTION poderia ser facilitada.

Após a aceitação do convite, os profissionais de TIC deveriam responder um questionário para verificação de perfil. Esses dados foram usados para divisão dos times. Vale ressaltar

que os profissionais convidados já se adequavam ao perfil mínimo especificado anteriormente.

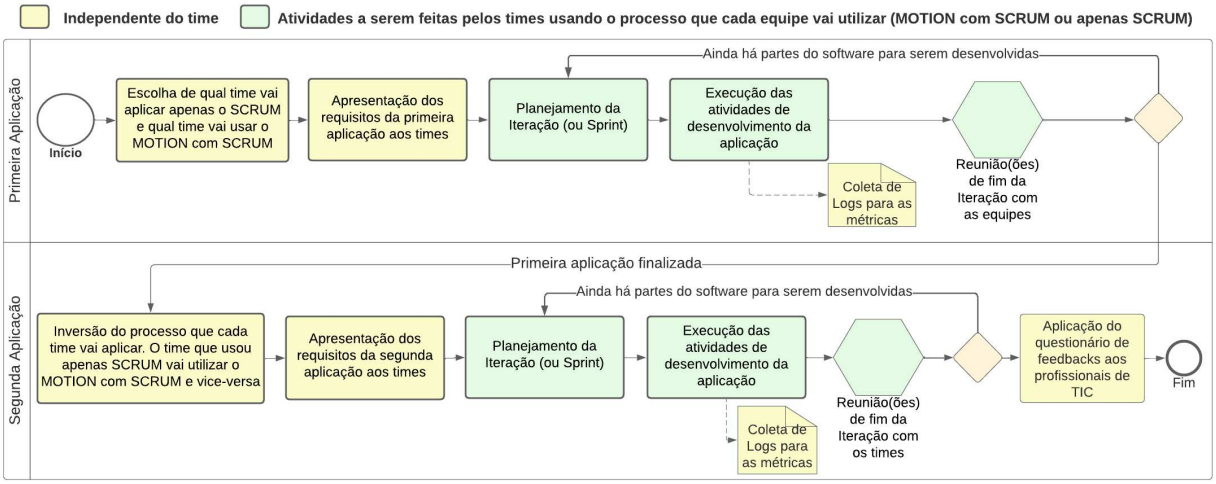
Na etapa seguinte, foi feita a explicação aos profissionais de como seria a avaliação e foi feita a divisão dos times considerando os perfis de cada profissional de TIC e sua disponibilização de tempo.

Por fim, na última etapa da preparação, foram feitos os treinamentos e distribuídos os materiais, incluindo a descrição do processo MOTION e seu Manual de Execução, a descrição e exemplos de implementação do modelo GRAFIT, o *template* ARTe e o *framework* KREATION, bem como seus manuais de uso e exemplos de aplicações desenvolvidas usando esses artefatos.

6.1.3.2 Planejamento da fase de execução e coleta de dados

A Figura 35 ilustra o planejamento da fase de execução e coleta de dados do experimento. Durante essa fase, os times de profissionais de TIC precisaram desenvolver duas aplicações. Os times desenvolveram as aplicações com base na mesma descrição informal fornecida pelo pesquisador, de modo que o pesquisador atuou como cliente, mas sem interferir diretamente nas decisões de cada time sobre o desenvolvimento das aplicações.

Figura 35 – Fase de execução e coleta de dados do experimento



Fonte: Próprio autor

A fase de execução e coleta de dados do experimento começa com a escolha da metodologia de desenvolvimento de software que será seguido por cada time (VALENTE, 2020).

O SCRUM é uma metodologia ágil conhecida que funciona bem com times pequenos, como no caso dos times desse experimento, e existem medidas próprias para avaliar o desempenho do time utilizando o SCRUM (DIXIT; BHUSHAN, 2019), que são necessárias nesse

experimento. Portanto, optou-se por utilizar o SCRUM juntamente com o processo MOTION para facilitar a avaliação e para garantir que certas medidas comuns durante o uso do SCRUM fossem verificadas.

O experimento então aconteceu de duas formas: com o uso do processo MOTION com a metodologia ágil SCRUM e com o uso da metodologia ágil SCRUM sem o MOTION. A cada ciclo de desenvolvimento de uma das aplicações, um time utilizou o Processo MOTION com SCRUM e o outro time utilizou apenas o SCRUM. Desse modo, pretendeu-se utilizar uma abordagem *cross-over* para o experimento, onde para cada aplicação um time funcionou como controle do outro.

É importante reforçar que o time que utilizou o MOTION com SCRUM na primeira aplicação, usou apenas o SCRUM na segunda aplicação, enquanto que o time que usou apenas SCRUM para a primeira aplicação, utilizou o MOTION com SCRUM para a segunda.

Um outro ponto que vale ressaltar é que o processo MOTION também pode ser utilizado junto com outras abordagens de desenvolvimento diferentes do SCRUM, inclusive processos não iterativos, como o processo cascata.

Após a escolha da metodologia para o desenvolvimento da aplicação IoHT autoadaptativa baseada em padrões de movimento, o pesquisador apresentou a descrição informal da primeira aplicação para ambos os times. Em seguida, começou o desenvolvimento propriamente dito por cada time.

Foi planejado que cada *sprint* (ou ciclo de desenvolvimento) durasse uma semana e cada ciclo deveria iniciar obrigatoriamente com uma reunião de planejamento da *sprint* e encerrar com a reunião de retrospectiva. Foi planejado que o pesquisador deveria acompanhar todas as reuniões para coleta de dados, mas não poderia interferir durante as mesmas, a não ser que fosse solicitado esclarecimentos sobre algum requisito, onde, nesse caso específico, o pesquisador (i.e., o autor da tese) atuaria como cliente, podendo responder a dúvida do time. Além das coletas de informações durante as reuniões, foram coletados dados do desenvolvimento através de *logs*. Ao fim do experimento, os profissionais de TIC responderam um questionário de *feedback*, cujas questões foram elaboradas com base no TAM (SILVA, 2015; MARIKYAN; PAPAGIANNIDIS, 2022).

Para versionamento de código, ambos os times utilizaram repositórios no Github[1] criados pelo pesquisador. Com base no código gerado pelos times e disponibilizado no Github, o

[1] https://github.com/

pesquisador e autor dessa tese foi capaz de acompanhar a evolução do código das aplicações implementadas e coletar *logs* (e.g., quantas linhas de código geradas por semana). Coube a cada time de desenvolvimento escolher quais ferramentas de gerenciamento de projeto e qual estrutura de descrição de requisitos e atividades (estórias de usuário ou caso de uso) utilizar.

Uma vez finalizado o tempo de desenvolvimento da primeira aplicação, foi feita a troca dos processos de desenvolvimento e o pesquisador apresentou a descrição informal da segunda aplicação. Finalmente, as mesmas etapas do experimento seguidas durante o desenvolvimento da primeira aplicação foram replicadas para a segunda aplicação. Para o desenvolvimento de cada aplicação os times tiveram 3 semanas.

Como o experimento tem como objetivo avaliar o processo MOTION e não as aplicações, ao final do período de desenvolvimento de cada aplicação, os times deveriam apresentar a versão da aplicação desenvolvida até o momento, ainda que não estivesse completa, e um conjunto mínimo de entregáveis, a fim de garantir que as diversas etapas do desenvolvimento de *software* foram seguidas (i.e., a elicitação de requisitos, o projeto da aplicação, a implementação, os testes e o planejamento da autoadaptação). Sendo assim, o conjunto mínimo de entregáveis contemplava: os principais requisitos da aplicação, um esboço do projeto da aplicação, o código fonte implementado, um plano ou relatório de testes e as regras de adaptação.

Os detalhes de toda execução do experimento são apresentados na subseção 6.2, incluindo a preparação, desde o convite dos profissionais de TIC até o treinamento, e a execução de todas atividades de desenvolvimento seguindo o planejamento apresentado que foi apresentado.

6.1.4 Planejamento da fase de síntese e análise dos dados

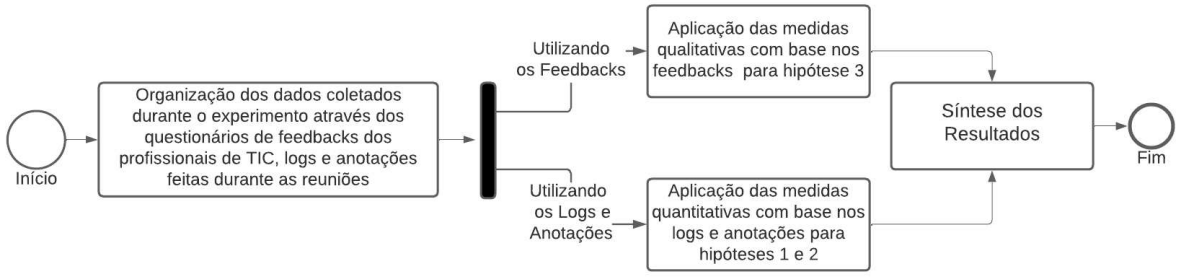
Nesta Seção são descritos o planejamento da atividade de síntese dos dados coletados durante o experimento e quais medidas foram aplicadas para a síntese dos dados e avaliação das hipóteses.

6.1.4.1 Síntese dos resultados

A Figura 36 apresenta o resumo do planejamento da atividade de síntese dos dados do experimento.

Essa atividade começa com a organização dos dados coletados durante o experimento de duas maneiras: através das resposta do questionário de *feedback*, respondido pelos profissionais de TIC, e da coleta de *logs* das aplicações e anotações dos *feedbacks* que o pesquisador

Figura 36 – Planejamento da fase de síntese dos resultados



Fonte: Próprio autor

obteve durante as reuniões de planejamento de *sprints*, reuniões de finalização de *sprints* e reuniões de retrospectiva.

Durante a etapa de organização, os dados coletados foram organizados em planilhas e foram feitos a verificação e o tratamento desses dados para identificar possíveis ambiguidades e removê-las. Por fim, foi feita a formatação desses dados, para ser possível aplicar as medidas.

Em paralelo, à medida que são formatados os dados, são feitas a aplicação das medidas quantitativas e qualitativas para verificação da hipótese 1, baseada nos dados obtido dos *logs* e das anotações do pesquisador, e a aplicação das medidas qualitativas para verificação das hipóteses 1, 2 e 3, baseada nos dados obtidos dos questionários de *feedbacks*. Por último, os resultados das medidas são sintetizados no documento final, para análise e discussão das hipóteses com base nesses resultados.

6.1.4.2 Medidas

Para avaliar as hipóteses, planejou-se medir os aspectos relacionados à percepção dos profissionais de TIC quanto à utilidade e facilidade de uso do processo MOTION, e usar medidas quantitativas conhecidas na literatura para avaliações da qualidade de uso de métodos ágeis, mais precisamente da metodologia SCRUM.

Para avaliação da utilidade e facilidade de uso, o autor da tese elaborou medidas com base no modelo TAM (SILVA, 2015; MARIKYAN; PAPAGIANNIDIS, 2022). As medidas usadas para avaliação do desempenho do time foram escolhidas com base em artigos identificados após uma busca na literatura por medidas de avaliação de projetos de *software* (DIXIT; BHUSHAN, 2019)(PEGORARO, 2014).

A Tabela 5 apresenta as medidas escolhidas para análise da hipótese 1.

As três primeiras medidas foram coletadas de (DIXIT; BHUSHAN, 2019), onde são

Tabela 5 – Medidas usadas para análise Hipótese 1

Medida	Como Coletar?	Como é medido?
Objetivo da *sprint*	*Logs* e *Feedbacks*	Percentual dos Objetivos da *sprint* = (nº de objetivos da *sprint* alcançados / nº total de objetivos da *sprint*)
Velocidade do Time	*Logs* e *Feedbacks*	nº de unidades de *software* desenvolvidas pelo time na *sprint*
Satisfação do Time	Questionários	Satisfação do time = (total de pontos para o time nas questões no questionário sobre satisfação durante o projeto / total de pontos do questionário multiplicado pela quantidade de membros do time)
Produtividade	*Logs* e *Feedbacks*	produtividade = (nº de linhas de código / tempo)
Estabilidade dos requisitos	*Logs* e *Feedbacks*	Estabilidade dos requisitos = (nº de requisitos inicial / nº total de requisitos)

Fonte: Próprio Autor

propostas medidas para avaliação de projetos desenvolvidos utilizando a metodologia SCRUM. Em especial, as medidas selecionadas se relacionam diretamente com o impacto que pode ser gerado pela utilização de um processo de desenvolvimento de *software* adequado.

O Objetivo da *Sprint* avalia o número de funcionalidades planejadas na *sprint* pelo número de funcionalidades implementadas.

Já a velocidade do time é uma medida que avalia o número médio de *story points* desenvolvidos em todas as *sprints*.

A terceira medida é a Satisfação do Time que indica o quanto o time se sente satisfeito com o desenvolvimento da aplicação. As oito perguntas utilizadas no questionário para a coleta da medida de Satisfação do Time estão na Tabela 6. As primeiras seis questões são referentes a Satisfação ao desenvolver cada aplicação, sendo metade exclusivamente para o desenvolvimento da primeira aplicação (questões Q1, Q3 e Q5) e a outra metade exclusivamente para o desenvolvimento da segunda aplicação (questões Q2, Q4 e Q6). Já as duas últimas correspondem a Satisfação durante todo o experimento. Os valores das respostas das questões Q1, Q3, Q5, Q7 e Q8 dos membros de cada time foram combinadas para obter o valor de Satisfação para primeira aplicação. E as respostas das questões Q2, Q4, Q6, Q7 e Q8 dos membros de cada time foram combinadas para obter o valor de Satisfação para segunda aplicação. Note que as questões Q7 e Q8 são consideradas para o cálculo da Satisfação do time nas duas aplicações por serem questões transversais que afetam o desenvolvimento de ambas as aplicações. Também é

calculado o valor total da Satisfação de cada time em relação ao desenvolvimento das aplicações durante todo o experimento.

Tabela 6 – Perguntas para análise de satisfação do time

	Questão sobre satisfação
Q1	Você está satisfeito com o seu desempenho durante o desenvolvimento da primeira aplicação?
Q2	Você está satisfeito com o seu desempenho durante o desenvolvimento da segunda aplicação?
Q3	Você está satisfeito com o desempenho do seu time durante o desenvolvimento da primeira aplicação?
Q4	Você está satisfeito com o desempenho do seu time durante o desenvolvimento da segunda aplicação?
Q5	Você está satisfeito com o processo de desenvolvimento de *software* usado pelo seu time durante o desenvolvimento da primeira aplicação?
Q6	Você está satisfeito com o processo de desenvolvimento de *software* usado pelo seu time durante o desenvolvimento da segunda aplicação?
Q7	Você está satisfeito com as tecnologias (linguagens de programação, *frameworks*, *templates*, bibliotecas, APIs) usadas pelo seu time durante o desenvolvimento das aplicações?
Q8	Você acredita que os processos utilizados durante o desenvolvimento das aplicações afetaram positivamente a satisfação com o seu desempenho e com o desempenho do seu time?

Fonte: Próprio Autor

As outras duas medidas da Tabela 5 foram coletadas de (PEGORARO, 2014) e são usadas para avaliar o desempenho em projetos de *software*. Elas foram coletadas pelos autores do artigo tanto através de análises da literatura quanto através de entrevistas com profissionais de TIC. A Produtividade indica a razão do número de linhas de código geradas pelo tempo. Já a Estabilidade de Requisitos é a razão do número de requisitos inicialmente planejados, pelo número total de requisitos.

A Tabela 7 apresenta as medidas escolhidas para análise da hipótese 2.

A primeira medida busca identificar a percepção de utilidade do processo MOTION pelos profissionais de TIC. Já a segunda medida busca avaliar a utilidade do Manual de uso e execução do processo MOTION, que é uma ferramente fundamental para a aplicação desse processo.

As perguntas utilizadas no questionário para a coleta das medidas de Percepção de utilidade do processo e Percepção de utilidade do manual do processo podem ser vistas respectivamente nas Tabelas 8 e 9. Além das perguntas apresentadas, para essas medidas, no

Tabela 7 – Medidas usadas para análise da Hipótese 2

Medida	Como Coletar?	Como é medido?
Percepção de utilidade do processo (baseada na medida Utilidade Percebida)	Questionários	Análise das questões do questionário sobre aderência ao processo. As questões terão respostas utilizando a escala Likert que varia de 1 a 5, onde 1 significa discordo plenamente e 5 significa concordo plenamente
Percepção de Utilidade do manual do processo (baseada na medida Utilidade Percebida)	Questionários	Análise das questões do questionário sobre aderência ao processo. As questões terão respostas utilizando a escala Likert que varia de 1 a 5, onde 1 significa discordo plenamente e 5 significa concordo plenamente

Fonte: Próprio Autor

questionário foi solicitado também, por sugestão dos especialistas de domínio que avaliaram o planejamento do experimento, alguns comentários dos profissionais de TIC sobre percepção de utilidade e facilidade de uso processo e de seu manual.

Tabela 8 – Perguntas para análise da medida de Percepção de utilidade do processo

	Questão
Q1	Usar o processo MOTION para guiar o desenvolvimento da aplicação permitiu realizar as tarefas mais rapidamente?
Q2	Usar o processo MOTION para guiar o desenvolvimento da aplicação melhorou seu desempenho na atividade?
Q3	Usar o processo MOTION para guiar o desenvolvimento da aplicação melhorou sua produtividade?
Q4	Usar o processo MOTION para guiar o desenvolvimento da aplicação aumentou sua eficácia no trabalho?
Q5	Usar o processo MOTION para guiar o desenvolvimento da aplicação tornou mais fácil fazer seu trabalho?
Q6	Você acredita que o processo MOTION é útil para seu trabalho?

Fonte: Próprio Autor

A Tabela 10 apresenta as medidas escolhidas para análise da hipótese 3.

A primeira medida busca avaliar a facilidade de aderência ao processo pela percepção dos profissionais de TIC. Já a segunda medida busca avaliar a percepção de facilidade de desenvolver aplicações IoHT autoadaptativas utilizando o processo MOTION.

As perguntas utilizadas no questionário para a coleta das medidas de Percepção de facilidade de aderência ao processo e Percepção de facilidade de desenvolver um sistema IoHT Autoadaptativo com o processo podem ser vistas, respectivamente, nas Tabelas 11 e 12.

Tabela 9 – Perguntas para análise da medida de utilidade do manual do processo

	Questão
Q1	Usar o manual do processo MOTION permitiu realizar as tarefas mais rapidamente?
Q2	Usar o manual do processo MOTION melhorou seu desempenho na atividade?
Q3	Usar o manual do processo MOTION melhorou sua produtividade?
Q4	Usar o manual do processo MOTION aumentou sua eficácia na atividade?
Q5	Usar o manual do processo MOTION tornou mais fácil fazer seu trabalho?
Q6	Você acha que o manual do processo MOTION é útil?

Fonte: Próprio Autor

Tabela 10 – Medidas usadas para análise da terceira hipótese

Medida	Como Coletar?	Como é medido?
Percepção de facilidade de aderência ao processo (baseada na medida Percepção de facilidade de uso)	Questionários	Análise das questões do questionário sobre aderência ao processo. As questões terão respostas utilizando a escala Likert que varia de 1 a 5, onde 1 significa discordo plenamente e 5 significa concordo plenamente
Percepção de facilidade de desenvolver um sistema IoHT Autoadaptativo com o processo (baseado na medida Percepção de facilidade de uso)	Questionários	Análise das questões do questionário sobre aderência ao processo. As questões terão respostas utilizando a escala Likert que varia de 1 a 5, onde 1 significa discordo plenamente e 5 significa concordo plenamente

Fonte: Próprio Autor

6.1.5 Planejamento da Análise dos Resultados

Os resultados das medidas selecionadas para análise e avaliação de cada hipótese permitem que a hipótese nula seja aceita ou rejeitada em detrimento da hipótese alternativa e fundamente a discussão final das questões de pesquisa do experimento.

As hipóteses nulas são rejeitadas em detrimento das hipóteses alternativas caso os resultados das medidas referentes as mesmas apresentem valores significativos que demonstrem que ao usar o processo MOTION foi possível haver melhora significativa em cada aspecto avaliado por cada medida. Essa significância dos resultados será obtida através de análise estatística, considerando a comparação entre os time e através da análise da percepção de utilidade e facilidade de uso do processo dos profissionais de TIC participantes do experimento.

A discussão dos resultados e hipóteses proveem embasamento para apresentação dos prós e contras do uso do processo MOTION e, com isso, é possível evidenciar se o processo é ou não viável e se ele é útil na percepção de profissionais de TIC.

Tabela 11 – Perguntas para análise da medida de Percepção de facilidade de aderência ao processo

	Questão
Q1	Aprender a usar o processo MOTION foi fácil para você?
Q2	Foi fácil utilizar o processo MOTION para guiar o desenvolvimento de uma aplicação?
Q3	Sua interação como processo MOTION para desenvolver um *software* foi clara e compreensível?
Q4	Você acha que o processo MOTION é claro e compreensível?
Q5	Você acha que foi fácil para você se tornar hábil ao usar o processo MOTION?
Q6	Você achou o processo MOTION fácil de usar?

Fonte: Próprio Autor

Tabela 12 – Perguntas para análise da medida de Percepção de facilidade de desenvolver um sistema IoHT autoadaptativo com o processo

	Questão
Q1	Aprender a usar o processo MOTION para desenvolver um sistema IoHT autoadaptativo foi fácil para você?
Q2	Foi fácil utilizar o processo MOTION para guiar o desenvolvimento de um sistema IoHT autoadaptativo como desejado?
Q3	Sua interação com o processo MOTION para desenvolver um sistema IoHT autoadaptativo foi clara e compreensível?
Q4	Você acha que as atividades presentes no processo MOTION para desenvolvimento de um sistema IoHT autoadaptativo são claras e compreensíveis?
Q5	Você acha que foi fácil para você se tornar hábil ao usar o processo MOTION para desenvolvimento de um sistema IoHT autoadaptativo?
Q6	Você achou o processo MOTION fácil de usar para desenvolvimento de um sistema IoHT autoadaptativo?

Fonte: Próprio Autor

6.2 Execução do experimento

Nesta Seção são detalhadas as atividades executadas durante as fases do experimento.

6.2.1 Fase de Preparação

Inicialmente foram convidados dez profissionais de TIC. Esses convites foram feitos por email e o contato, incluindo a confirmação dos participantes durou duas semanas. Dois dos profissionais convidados optaram por não participar do experimento devido a questões de agenda. Desse modo, oito profissionais de TIC participaram do experimento. Cada um desses profissionais de TIC assinou um Termo de Consentimento Livre Esclarecido (TCLE) concordando com os termos da pesquisa e se comprometendo em participar das atividades do

experimento. O modelo de TCLE que os profissionais de TIC assinaram pode ser visto no Apêndice B.

A Tabela 13 apresenta o perfil dos profissionais de TIC participantes. Os profissionais de TIC estão representados com identificadores começando pela letra "P". Vale ressaltar que participaram do experimento duas profissionais de TIC do sexo feminino (as profissionais P5 e P7) e seis profissionais de TIC do sexo masculino (os profissionais P1, P2, P3, P4, P6 e P8). O questionário aplicado para coleta dos dados de perfil do usuário foi gerado e disponibilizado através da ferramenta Google Forms[2] e está disponível para acesso em https://bit.ly/44OR851.

Tabela 13 – Perfil dos participantes do experimento

Experiência com	P1	P2	P3	P4	P5	P6	P7	P8
desenvolvimento de software	2 a 5 anos	5 a 10 anos	2 a 5 anos	2 a 5 anos	2 a 5 anos	1 a 2 anos	2 a 5 anos	Mais de 10 anos
desenvolvimento de aplicações móveis	2 a 5 anos	1 a 2 anos	1 a 2 anos	1 a 2 anos	1 a 2 anos	1 a 2 anos	1 a 2 anos	5 a 10 anos
o uso de metodolgias ágeis, incluindo SCRUM	2 a 5 anos	2 a 5 anos	2 a 5 anos	2 a 5 anos	2 a 5 anos	Menos de 1 ano	2 a 5 anos	5 a 10 anos
desenvolvimento de sistemas autoadaptativos ou sensíveis a contexto	1 a 2 anos	Não possui	Não possui	Menos de 1 ano	Não possui	Não possui	Não possui	Mais de 5 anos
desenvolvimento de sistemas IoT	1 a 2 anos	1 a 2 anos	Não possui	Não possui	Não possui	Menos de 1 ano	Menos de 1 ano	Mais de 5 anos
desenvolvimento de sistemas de saúde	1 a 2 anos	1 a 2 anos	Não possui	Não possui	Menos de 1 ano	Menos de 1 ano	Menos de 1 ano	Não possui

Fonte: Próprio Autor

Após a escolha dos participantes foi feita a apresentação dos detalhes do experimento e foram divididos os times. Para escolher os integrantes de cada time foi feita uma análise do perfil e também foi preciso considerar a disponibilidade de tempo de cada um para execução das atividades, de modo que fosse possível haver ao menos um encontro semanal entre o pesquisador e os times completos. O primeiro time foi integrado pelos profissionais P3, P4, P5 e P6. Já os profissionais P1, P2, P7 e P8 compuseram o segundo time.

Note que a proporção de profissionais do sexo masculino e feminino foi a mesma para cada time e que todos os profissionais de TIC de ambos os times possuem experiência de mais de 1 ano com desenvolvimento de aplicações móveis e com o uso de metodologias ágeis, como o SCRUM. Contudo, um número maior de profissionais com conhecimento em aplicações sensíveis ao contexto e aplicações de saúde compuseram o segundo time. Esse desbalanceamento aconteceu devido as limitações de horários dos participantes. Porém, foi garantido que em ambos os times houvesse ao menos dois profissionais TIC com algum conhecimento em

[2] https://docs.google.com/forms

desenvolvimento de aplicações de saúde e ao menos um profissional TIC com conhecimento no desenvolvimento de aplicações sensíveis ao contexto.

Os profissionais de TIC foram então treinados utilizando a estratégia de sala de aula invertida, onde primeiro os materiais de estudo, contemplando vídeos, slides e documentação, são fornecidos aos profissionais de TIC e, posteriormente, é marcado um encontro para retirada de dúvidas. Esse encontro aconteceu cinco dias após o envio dos materiais dos treinamentos em 10 de maio de 2023.

Foram gravados quatro vídeos de treinamentos contemplando os temas: (i) Fundamentação Teórica sobre sistemas de *Internet of Health Things* e sistemas autoadaptativos; (ii) O processo MOTION e como utilizá-lo; (iii) O modelo de grafo de classificação GRAFIT; e (iv) *Framework KREATION* e *template* ARTe. Os vídeos com os treinamentos estão disponíveis em https://bit.ly/3DhPSvD.

6.2.2 Fase de Execução

O experimento foi executado no período entre 17 de Maio e 30 de Junho de 2023 (6 semanas e 2 dias). A primeira aplicação foi desenvolvida pelos times no período de 17 de maio a 6 de Junho. Enquanto a segunda foi desenvolvida entre os dias 7 e 30 de junho. Nesse período, o pesquisador se reuniu com cada um dos times em todas as semanas utilizando a ferramenta Google Meet [3]. A cada reunião eram definidas as atividades das *sprints* semanais de cada time, bem como era feita a retrospectiva da *sprint* da semana anterior. Com o consentimento dos profissionais de TIC, todas as reuniões foram gravadas pelo pesquisador, para posterior revisão e coleta de dados.

As descrições informais das aplicações foram enviadas em formato de vídeo aos times na primeira semana (para primeira aplicação) e na quarta semana (para a segunda aplicação). Nas reuniões das duas semanas supracitadas, a descrição das aplicações também foram reforçadas pelo pesquisador. Os papéis de cada participante do experimento eram definidos internamente entre os membros dos times de desenvolvimento. Para cada time também foi criado um repositório na plataforma github, onde deveriam dispor os códigos das aplicações.

A seguir são apresentadas as descrições informais das duas aplicações.

Para a primeira aplicação foi instruído que o primeiro time deveria utilizar o processo MOTION com SCRUM, enquanto que o segundo time deveria utilizar apenas a metodologia

[3] https://meet.google.com/

SCRUM.

Descrição informal da primeira aplicação
Eu gostaria de uma aplicação capaz de detectar a ocorrência de quedas usando os sensores de acelerômetro e giroscópio do dispositivo Android. Quando uma possível queda for detectada, a aplicação deve lançar um alerta, perguntando se o usuário está bem. Caso a pessoa não responda em 10s, ou caso solicite ajuda, a aplicação deve enviar um alerta de queda para uma pessoa (cuidador/parente/profissional de saúde) cadastrada, informando a localização do usuário. A aplicação deve se adaptar para garantir economia de bateria quando necessário.

Para a segunda aplicação foi instruído que o segundo time utilizasse o processo MOTION com SCRUM, enquanto que o primeiro time deveria utilizar apenas a metodologia SCRUM. Além disso, ao invés de informar o contexto que deveria ser utilizado para a autoadaptação, como na primeira aplicação, cada time deveria definir e implementar algum tipo de autoadaptação (vantajosa) para o *app*.

Descrição informal da segunda aplicação
Eu gostaria de uma aplicação Android capaz de detectar a ocorrência de noctúria e monitorar a qualidade de sono. A qualidade de sono pode ser inferida com base na análise do período de sono e é um tipo de informação coletada normalmente por *smartwatches* ou *smartbands* (vocês podem usar a API Google Fit para coletar esses dados). Já a noctúria, também chamada de diurese noturna, é uma vontade de urinar frequente que acontece durante à noite e faz com que o paciente tenha que interromper seu sono, diversas vezes, para poder aliviar a vontade de urinar. Caso seja detectada a noctúria em períodos contínuos de 2 a 3 dias, o paciente deve ser alertado.

Para ambas as aplicações, os dois times de desenvolvimento também puderam utilizar os artefatos de reúso de *software* propostos nessa tese como apoio ao desenvolvimentos das aplicações. Outras ferramentas também poderiam ser utilizadas à escolha de cada time.

A última reunião com os times de desenvolvimento foi feita na primeira semana de julho de 2023. Nessa reunião, foi feita a última retrospectiva e os profissionais de TIC responderam o questionário final de *feedbacks* para coleta dos dados referentes as medidas de Percepção de Facilidade, Percepção de Utilidade e Satisfação do Time. Esse questionário também foi gerado e disponibilizado com a ferramenta Google Forms e pode ser acessado através do link: https://bit.ly/3pTNBnj.

Devido a questões pessoais, o profissional de TIC P2 contribuiu pouco para o desenvolvimento da segunda aplicação, justamente quando seu time utilizou o processo. Segundo o mesmo, isso impactou nas respostas do questionário de *feedbacks*, especialmente as questões

relativas a Satisfação do Time. Desse modo, na próxima seção, os resultados são apresentados com e sem as respostas desse profissional de TIC para a medida de Satisfação do Time.

6.3 Resultados do Experimento

Os resultados das medidas foram sintetizados usando uma planilha no Google Forms. Os dados coletados e sintetizados do experimento estão disponíveis em https://bit.ly/44UOYRF

As medidas para avaliação da primeira hipótese foram calculadas para cada time considerando os requisitos descritos, objetivos planejados, código implementado, funcionalidades e artefatos produzidos para cada aplicação e a satisfação dos membros do time. As medidas para avaliação da segunda e terceira hipóteses consideraram a perspectiva de todos os profissionais de TIC participantes do experimento.

Para análise estatística dos resultados foi considerada a significância crítica (*p-value*) de 5% para rejeição de qualquer uma das hipóteses nulas em detrimento das hipóteses alternativas. Destaca-se que os cálculos foram feitos com o apoio da ferramenta SPSS Statistics (STATISTICS, 2012). A seguir são apresentados e detalhados os resultados obtidos com o experimento.

6.3.1 Resultados das medidas para primeira hipótese

Para a primeira hipótese foram coletadas as medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade, Estabilidade dos requisitos e Satisfação do Time para cada *sprint*. Essas medidas foram coletadas para cada time e para cada *sprint* do desenvolvimento de cada aplicação.

Os resultados sintetizados para as medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade dos requisitos para o time 1 estão na Tabela 14. Enquanto os resultados para o time 2 estão na Tabela 15.

Para o desenvolvimento de cada aplicação, os times executaram três *sprints* de uma semana cada. Desse modo foi obtida uma amostra de tamanho 6 (3 para cada aplicação) para análise das medidas: Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade dos requisitos. Para Satisfação do Time foi calculado um valor para cada membro dos times e para cada aplicação. Assim foi obtida uma amostra de tamanho 8 (4 para cada aplicação) para análise dessa medida. Devido a número muitíssimo pequeno de amostras e por parte das medidas não seguir uma distribuição normal, a análise estatística da hipótese 1 foi feita utilizando testes

Tabela 14 – Resultados das medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade dos requisitos para o Time 1

Medida	Aplicação	*Sprint* 1	*Sprint* 2	*Sprint* 3	Média
Objetivo da *Sprint*	1	0,67	1	0,71	0,79
Velocidade do Time	1	1	3	5	3
Produtividade	1	0	4	6,4	3,47
Estabilidade dos requisitos	1	0,88	0,88	0,88	0,88
Objetivo da *Sprint*	2	0,33	0,5	0,6	0,5
Velocidade do Time	2	1	3	3	2,33
Produtividade	2	0	2	4,4	2,13
Estabilidade dos requisitos	2	1	1	1	1

Fonte: Próprio Autor

não paramétricos como indicado em (LAUREANO *et al.*, 2020; CONTADOR; SENNE, 2016; SIEGEL; JR, 2006).

Tabela 15 – Resultado das medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade dos requisitos para o Time 2

Medida	Aplicação	*Sprint* 1	*Sprint* 2	*Sprint* 3	Média
Objetivo da *Sprint*	1	0,67	0,75	0,80	0,74
Velocidade do Time	1	2	5	2	3
Produtividade	1	98	15,6	31,2	48,27
Estabilidade dos requisitos	1	0,7	0,7	0,7	0,7
Objetivo da *Sprint*	2	0,67	0,75	0,67	0,7
Velocidade do Time	2	1	3	5	3
Produtividade	2	0	7	9	5,33
Estabilidade dos requisitos	2	1	1	1	1

Fonte: Próprio Autor

Para a aplicação 1, pode-se observar que o time 1 obteve resultados similares aos do time 2 para as medidas de Objetivo da *Sprint* e Velocidade do time. Já o time 2 apresentou melhor Produtividade. A diferença para essa medida durante o desenvolvimento da aplicação 1 foi grande, porém, isso se deve ao fato de o time 2 desenvolver o sistema sem utilizar o *framework* KREATION para a primeira aplicação. Além disso, um dos membros do time 2 já havia desenvolvido uma aplicação similar antes. Mesmo assim, os resultados de funcionalidades das aplicações desenvolvidas por cada time para a aplicação 1 foram similares, pois segundo o relado do time 1, o *framework* KREATION facilitou o desenvolvimento da maioria das funcionalidades dessa aplicação. Em relação a medida de Estabilidade dos requisitos, percebe-se que o time 1 precisou modificar menos os requisitos entre o começo e o fim do projeto do que o time 2.

Acredita-se que isso ocorreu devido ao time 1 ter se dedicado apenas aos requisitos na primeira *sprint*, enquanto o time 2 dividiu sua atenção com o início do desenvolvimento da aplicação. Vale destacar que, para o desenvolvimento dessa aplicação, apenas o time 1 utilizou o processo MOTION.

Para a aplicação 2, pode-se observar que o time 2 obteve resultados similares aos do time 1 para as medidas de Produtividade e Estabilidade dos Requisitos. Dessa vez, ambos os times utilizaram o *framework* KREATION e optaram por trabalhar apenas com requisitos na primeira *sprint* e codificar a partir da segunda. O time 2 apresentou melhor desempenho para as demais medidas. Vale ressaltar que o time 2 utilizou o processo MOTION para o desenvolvimento da segunda aplicação. Além disso, um dos membros do time 2 não conseguiu contribuir para o desenvolvimento dessa aplicação, como já foi informado.

Em termos ordinais é percebido resultados melhores para as medidas de Objetivo da *Sprint*, Velocidade do Time e Produtividade para time 1 quando utilizou o processo MOTION para desenvolver a aplicação em comparação quando não utilizou o processo. No caso do time 1, em termos ordinais, é percebido resultados similares para as medidas de Objetivo da *Sprint* e Velocidade do Time para as duas aplicações e melhores para Estabilidade de Requisitos quando utilizou o processo MOTION. Note que para o time 2 que não utilizou o *framework* KREATION para a primeira aplicação, mas que utilizou para a segunda, a produtividade possui uma grande diferença, mesmo tendo o time entregue quantidades similares de funcionalidades para as duas aplicações.

A Tabela 16 apresenta os resultados da análise estatística para as medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade, Estabilidade dos requisitos. O teste foi feito utilizando a ferramenta SPSS Statiscics [4] que retorna um valor de significância apenas para análise bicaudal para todos os testes não paramétricos, porém, para Hipótese 1 é desejada uma avaliação unicaudal, pois com a hipótese alternativa espera-se obter uma melhora, enquanto para a hipótese nula é considerada a situação em que não há melhora, ou há uma piora. Então, ao invés da significância informada pela ferramenta, obteve-se o valor Z estatístico, também calculado pela ferramenta. Com esse valor foi possível obter o p-value adequado para cada teste presente na Tabela 16.

Foi feita uma análise de variáveis relacionadas comparando o desempenho de cada time para cada aplicação (tipo de análise A1) usando o teste não paramétrico de Wilcoxon, como

[4] A SPSS Statistics está disponível para download em: https://www.ibm.com/br-pt/spss

Tabela 16 – Análise estatística para as medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade dos requisitos

Medida	Time(s)	Tipo de Análise	Valor de Z	P-value
Objetivo da *Sprint*	T1	A1	1,64	0,05
Velocidade do Time	T1	A1	1,00	0,16
Produtividade	T1	A1	1,41	0,079
Estabilidade dos requisitos	T1	A1	1,73	**0,04**
Objetivo da *Sprint*	T2	A1	1,00	0,16
Velocidade do Time	T2	A1	0	0,5
Produtividade	T2	A1	1,60	0,05
Estabilidade dos requisitos	T2	A1	1,73	**0,04**
Objetivo da *Sprint*	T2 e T1	A2	0	0,5
Velocidade do Time	T2 e T1	A2	0	0,5
Produtividade	T2 e T1	A2	1,96	**0,02**
Estabilidade dos requisitos	T2 e T1	A2	2,23	**0,01**
Objetivo da *Sprint*	T1 e T2	A3	1,99	**0,02**
Velocidade do Time	T1 e T2	A3	0,64	0,26
Produtividade	T1 e T2	A3	0,88	0,18
Estabilidade dos requisitos	T1 e T2	A3	0	0,5

Fonte: Próprio Autor

feito em (LAUREANO *et al.*, 2020; SIEGEL; JR, 2006). As variáveis são relacionadas, pois foi analisado o desempenho do mesmo grupo de indivíduos (time) para o desenvolvimento de diferentes aplicações. Além disso, foi feita uma análise comparando as medidas entre o time 1 e o time 2 para o desenvolvimento da primeira aplicação (tipo de análise A2) e entre time 2 e o time 1 para o desenvolvimento da segunda aplicação (tipo de análise A3). Nessas duas últimas análises, foi comparado o desempenho de indivíduos de times diferentes durante o desenvolvimento de cada aplicação, logo, optou-se por utilizar um método não paramétrico para análise de variáveis independentes. O método utilizado para essas análises foi o de Mann-Whitney, o mesmo utilizado em (CONTADOR; SENNE, 2016; CASTRO, 2012; SIEGEL; JR, 2006).

Para avaliação feita comparando o desempenho dos times ao desenvolver a primeira e a segunda aplicação, apenas foi percebido um valor significante (p-value < 0,05) para a medida de Estabilidade de requisitos, sendo que para o time 1 a Estabilidade de requisitos foi menor quando utilizou o processo MOTION, enquanto que foi maior para o time 2 quando utilizou esse processo. Porém, nas análises onde são comparados os times ao desenvolver as aplicações, percebe-se valores significantes para as medidas de Produtividade e Estabilidade de requisitos para o desenvolvimento da primeira aplicação, e a medida de Objetivo da *Sprint* para o desenvolvimento da segunda aplicação. Com isso, não foram obtidos resultados suficientes

para rejeitar a Hipótese nula 1, considerando os aspectos avaliados pelas medidas de Objetivo da *Sprint*, Velocidade do Time, Produtividade e Estabilidade de Requisitos.

Um resumo das respostas das perguntas referentes a medida de Satisfação do Time para cada time pode ser visto na Figura 37.

Figura 37 – Medida de Satisfação dos time. Os valores Q1 a Q8 representam as questões do questionário de satisfação

(a) Satisfação para o time 1 (b) Satisfação para o time 2

Fonte: Próprio autor

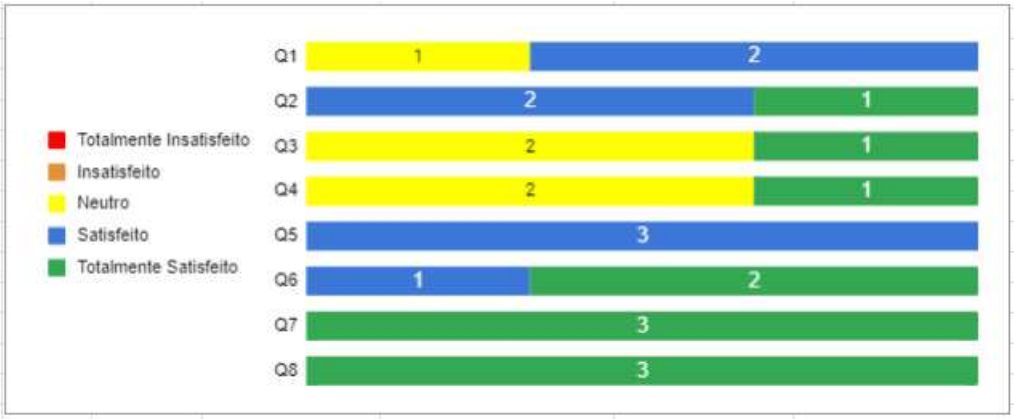
É possível perceber que para as questões ligadas ao desenvolvimento da primeira aplicação (Q1, Q3, Q5, Q7 e Q8), o time 1 se mostrou, em geral, satisfeito com o desempenho durante o desenvolvimento dessa aplicação, com apenas um membro do time se mostrando neutro para a questão Q1 (“Você está satisfeito com o seu desempenho durante o desenvolvimento da primeira aplicação?”). Enquanto que o time 2 se mostrou mais Satisfeito que Insatisfeito, com apenas um membro do time se mostrando insatisfeito com a questão Q5 (“Você está satisfeito com o processo de desenvolvimento de *software* usado pelo seu time durante o desenvolvimento da primeira aplicação?”).

Para as questões ligadas ao desenvolvimento da segunda aplicação (Q2, Q4, Q6, Q7 e Q8), o time 1 se mostrou mais satisfeito que insatisfeito, com apenas uma menção de insatisfação do time para a questão Q4 (Você está satisfeito com o desempenho do seu time durante o desenvolvimento da segunda aplicação?). Vale destacar também que, para a questão Q6 (Você está satisfeito com o processo de desenvolvimento de *software* usado pelo seu time durante o desenvolvimento da segunda aplicação?), o time 1 se mostrou mais neutro.

Já o time 2, considerando todos o membros do time, se mostrou um pouco mais satisfeito que insatisfeito com o desenvolvimento da aplicação 2. Sendo que um membro do time se mostrou totalmente insatisfeito com relação a questão Q2 (Você está satisfeito com o seu desempenho durante o desenvolvimento da segunda aplicação?) e insatisfeito em relação a questão Q6. Note também que o time 2 se mostrou mais neutro que satisfeito em relação a

questão Q4.

Figura 38 – Satisfação para o time 2 sem *outlier*



Fonte: Próprio Autor

Como mencionado anteriormente, um dos membros do time 2 declarou que não conseguiu contribuir no desenvolvimento da segunda aplicação por questões externas ao experimento, o que principalmente impactou na medição de satisfação para esse membro. Quando retiradas as respostas desse profissional de TIC, foi obtido o resultado da Figura 38. Assim é possível perceber que não há grande impacto na satisfação do time 2 em relação ao desenvolvimento da primeira aplicação, onde foram consideradas as questões (Q1, Q3, Q5, Q7 e Q8), mas o time 2 se mostra mais satisfeito em relação ao desenvolvimento da segunda aplicação.

A Tabela 17 apresenta os valores da medida de Satisfação do Time para cada time por aplicação desenvolvida. Esses valores foram calculados com base nas respostas do questionário. E também é apresentada a satisfação geral do time, considerando as respostas de todas as questões pelos membros de cada time. Importante destacar que a satisfação geral do time não é uma média da satisfação do time para cada aplicação, pois para isso iria ter de considerar duas vezes as respostas das questões Q7 e Q8, que são medidas para a satisfação do time em cada aplicação.

Pela Tabela 17, é possível observar que, de um modo geral, a satisfação de ambos os times ficaram acima de 75% para o desenvolvimento de ambas as aplicações, o que é considerada alta segundo (DIXIT; BHUSHAN, 2019). Para o time 1, a medida de Satisfação do Time foi maior durante o desenvolvimento da aplicação 1, quando utilizaram também o processo MOTION. Já para o time 2, ao serem considerados todos os membros, a Satisfação do Time se manteve a mesma para ambas as aplicações, mas quando retirado o valor do *outlier* (valor do membro do time que não conseguiu contribuir para o desenvolvimento da segunda aplicação), o

Tabela 17 – Resultado das medidas de Satisfação do Time (T1 = Time 1, T2 = Time 2, Total T1 = Total para time 1 e Total T2 = Total para time 2)

Time	Aplicação	Com MOTION?	Satisf. do Time	Satisf. do Time sem *outlier*
T1	1	Sim	0,9	-
T1	2	Não	0,83	-
Total T1	1 e 2	-	0,85	-
T2	1	Não	0,82	0,85
T2	2	Sim	0,82	0,91
Total T2	1 e 2	-	0,79	0,85

Fonte: Próprio Autor

valor da medida de Satisfação do Time é maior durante o desenvolvimento da segunda aplicação, quando utilizaram também o processo MOTION.

Foi feita a análise estatística considerando a comparação dos resultados de Satisfação dos times para cada aplicação, a fim de verificar se é ou não possível rejeitar a hipótese nula 1, considerando apenas o aspecto da satisfação. Os resultados podem ser vistos na Tabela 18. A análise foi feita usando também o teste de Wilcoxon com auxílio da ferramenta SPSS Statistics.

Tabela 18 – Análise estatística para a medida de Satisfação do Time comparando o desenvolvimento das duas aplicações

Time	Valor Z	Valor Z sem *outlier*	p-value	p-value sem *outlier*
T1	1,34	-	0,09	-
T2	0	1,34	0,5	0,09

Fonte: Próprio Autor

Para o time 1 foi obtido um valor de significância de 0,09. Já para o time 2 foram feitas duas análises, uma com os quatro membros do time e outra com apenas três, retirando os resultados do membro do time 2 que relatou que não conseguiu se dedicar ao experimento durante o desenvolvimento da segunda aplicação. Para o primeiro caso foi obtido um valor de significância de 0,5 e para o segundo caso foi obtido o valor 0,09. Em todos os casos, o valor de significância foi superior a 5%. Desse modo também não é possível rejeitar a hipótese nula 1 considerando o aspecto avaliado pela medida de Satisfação do Time.

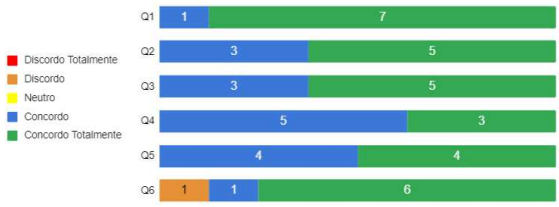
Não foi feita a comparação entre os times para a medida de Satisfação do Time, pois foi entendido que o impacto dessa comparação seria pouco e seria afetado pelo resultado da satisfação do membro do time 2 que não conseguiu se dedicar completamente durante o desenvolvimento da segunda aplicação.

6.3.2 Resultados das medidas para segunda hipótese

A Figura 39 apresenta um resumo das respostas do questionário de *feedback* contendo as questões selecionadas relativas as duas medidas usadas para avaliar a percepção de utilidade do processo.

Como pode ser observado na Figura 39a, na percepção de 100% dos profissionais de TIC que participaram do experimento, o uso do processo MOTION para guiar o desenvolvimento de aplicações permite a realização de tarefas mais rapidamente (Q1), podendo melhorar o desempenho (Q2), produtividade (Q3) e eficácia (Q4) do profissional nas atividades de desenvolvimento. Além disso, todos os profissionais de TIC consideram que o uso do processo MOTION é útil para tornar mais fácil o desenvolvimento das aplicações IoHT autoadaptativas baseadas em padrões de movimento.

Figura 39 – Resultados das questão sobre percepção de utilidade. Os valores Q1 a Q6 representam as questões do questionário



(a) Percepção de Utilidade de uso do processo MOTION

(b) Percepção de Utilidade do Manual de uso do processo MOTION

Fonte: Próprio autor

No entanto, enquanto 87% dos profissionais de TIC participantes do experimento acreditam que o uso do processo MOTION é útil para seu trabalho, apenas um dos profissionais de TIC não concorda com essa afirmação. Vale destacar que esse profissional de TIC relatou que atualmente não trabalha com o desenvolvimento de aplicações IoHT ou autoadaptativas. Além disso, esse é o mesmo profissional que participou do time 2 e que informou que não conseguiu atuar durante o desenvolvimento da segunda aplicação, justamente quando o seu time utilizou o MOTION como guia para auxiliar o desenvolvimento da aplicação, ainda assim ele identificou o processo MOTION como útil baseado em sua experiência prática.

Observando a Figura 39b, é possível ver que todos os profissionais de TIC também concordaram que o manual de uso do processo MOTION é útil para guiar a execução de tarefas mais rapidamente (Q1). Além disso, na percepção de todos os profissionais de TIC, o manual se mostrou útil para melhorar o desempenho (Q2), produtividade (Q3) e eficácia (Q4) dos

profissionais durante o desenvolvimento das aplicações. Os profissionais também concordaram que o manual de uso do processo MOTION é útil. Por fim, 87% dos participantes do experimento consideram que o manual de uso e execução do processo MOTION tornou mais fácil o trabalho, enquanto que um dos participantes se mostrou neutro em relação a essa questão.

Para análise estatística da segunda hipótese, foram calculados os valores médios das medidas de Percepção de utilidade do processo MOTION (PUP) e Percepção de utilidade do manual de uso e execução do processo MOTION (PUM) para cada profissional de TIC (Tabela 19) e para cada questão (Tabela 20).

Tabela 19 – Medidas da avaliação de Utilidade por profissional de TIC

Profissional de TIC	Média PUP	Média PUM
P1	5,0	5,0
P2	3,8	4,00
P3	4,3	4,50
P4	4,7	4,67
P5	4,5	4,50
P6	4,3	4,17
P7	5,0	5,00
P8	5,0	5,00

Fonte: Próprio Autor

Para análise e escolha do teste estatístico a ser utilizado na verificação dessa hipótese, observou-se que o tamanho da amostra é muito pequeno, porém não é muito diferente da utilizada com o modelo TAM (SILVA, 2015; MARIKYAN; PAPAGIANNIDIS, 2022) para outros estudos de avaliação de novos artefatos de *software*, como em (ARAÚJO, 2022), logo, não podia ser descartada a vantagem do uso de uma estratégia paramétrica, caso a distribuição da amostra fosse configurada como normal. A fim de verificar a normalidade da distribuição da amostra para o teste t (MARTINEZ; FERREIRA, 2007), foi utilizada a ferramenta SPSS Statistics (STATISTICS, 2012) e o teste de normalidade de Kolmogorov-Smirnov (BERGER; ZHOU, 2014), que comprovou a normalidade na distribuição dos dados. De maneira similar também foi verificada a normalidade dos dados para as medidas referentes a hipótese 3.

Com isso, foi decidido que seria utilizado o teste t para análise estatística dos dados. No entanto, ainda considerando o possível viés decorrente do tamanho da amostra, também foi utilizado o teste não paramétrico de Wilcoxon. Para o teste t, as amostras foram verificadas em comparação ao valor 3, correspondente ao valor médio para resposta das questões, referente a neutralidade da escala de Likert. Para o teste de Wilcoxon, a comparação foi feita

Tabela 20 – Medidas da avaliação de Utilidade por questão

Questão	Média PUP	Média PUM
Q1	4,88	4,50
Q2	4,63	4,50
Q3	4,63	4,63
Q4	4,38	4,63
Q5	4,50	4,38
Q6	4,50	5,00

Fonte: Próprio Autor

utilizando o valor 3 da mediana. Importante destacar que foi verificada a significância para o teste unicaudal, já que na hipótese alternativa é questionado se o processo MOTION é útil para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento na percepção dos profissionais de TIC. Os resultados da análise das medidas para os dois testes pode ser visto na Tabela 21.

Tabela 21 – Análise estatística das medidas de utilidade

Medida	Referência	teste t	Wilcoxon
PUP	Valor por profissional de TIC	inferior à 0,01	0,005
PUM	Valor por profissional de TIC	inferior à 0,01	0,005
PUP	Valor por questão	inferior à 0,01	0,013
PUM	Valor por questão	inferior à 0,01	0,013

Fonte: Próprio Autor

Para o teste t, a ferramenta SPSS Statistics calculou um valor muito pequeno e retornou como resultado que o valor é menor que 1% para as duas medidas considerando a análise da média por profissionais de TIC e por questão. Para o teste de Wilcoxon, também foram obtidos valores inferiores ao valor de 5% da significância crítica (p-value) para as duas medidas. Desse modo, em todos os cenários avaliados é possível rejeitar a hipótese nula 2 em detrimento da hipótese alternativa 2.

Além das questões elaboradas com base no modelo TAM, para coleta e análise da percepção utilidade do processo MOTION e de seu manual de uso, também foram coletados comentários adicionais dos profissionais de TIC participantes do experimento. Vale destacar que os comentários foram fornecidos de maneira opcional e nem todos os profissionais de TIC apresentou comentários.

Os comentários coletados sobre a percepção de uso do processo MOTION então na Tabela 22. Os comentários reforçam a percepção de utilidade do processo MOTION pelos

profissionais de TIC participantes do experimento e indicam que é vantajoso o uso de processos de desenvolvimento de *softwares* voltados à domínios específicos.

Tabela 22 – Comentários sobre a experiência dos profissionais de TIC com o uso do processo MOTION

Comentário
"A utilização do processo MOTION auxiliou no desenvolvimento da aplicação além de que facilitou no desenvolvimento da aplicação proposta."
"Já tinha trabalhado em desenvolvimento de aplicações, no entanto, tinha grande dificuldade no processo de desenvolvimento, uma vez que não tinha um processo que funcionasse como guia do desenvolvimento. Com o MOTION, esse problema foi resolvido, deixando o processo de desenvolvimento mais fácil de entender e por em prática."
"O processo Motion deixa explícito muitos dos passos necessários para desenvolver uma aplicação IoHT, forçando o time a deixar explícito coisas como componentes regras de adaptação, saber desses detalhes antes de começar a implementação agiliza a desenvolvimento, o que se percebeu no desenvolvimento sem o processo Motion foi a falta de clareza do que estava sendo desenvolvido, e como seria desenvolvido."
"Devido a pouca experiência com desenvolvimento de aplicações voltadas a IOT, o processo MOTION foi de grande valia para que eu pudesse entender o que deveria ser feito e como realizar cada atividade."

Fonte: Próprio Autor

A Tabela 23 contém os comentários sobre o uso do manual do processo MOTION. É possível perceber que os profissionais realmente utilizaram o manual e o consideraram útil para auxiliar o desenvolvimento das aplicações. Vale destacar que no terceiro comentário foi apresentada uma sugestão para melhoria do manual de uso do processo MOTION. Essa sugestão deve ser incorporada em uma versão futura após novo refinamento do processo.

Finalmente, a Tabela 24 apresenta comentários sobre o desenvolvimento das aplicações IoHT autoadaptativas. Com base nos comentários dessa Tabela é possível perceber que alguns dos profissionais de TIC observaram dificuldades no desenvolvimento da parte de coleta de dados, e que, nesse caso, artefatos de *softwares* específicos para esse tipo de domínio, como o *framework* KREATION, contribuem para o desenvolvimento dessa aplicação tanto para pessoas com experiência no domínio da aplicação, quanto para pessoas sem essa experiência.

Também destaco o último comentário da Tabela 24, que reforça como as sugestões fornecidas pelo processo MOTION e seu manual auxiliaram o desenvolvimento da aplicação mesmo para um profissional sem experiência com aplicações IoHT.

Tabela 23 – Comentários sobre a experiência dos profissionais de TIC com o uso do manual do processo MOTION

Comentário
“O manual foi útil para conseguir um detalhamento do processo antes de utilizá-lo. Em seguida, no próprio desenvolvimento da aplicação também foi útil visto que já tinha uma base do processo à ser utilizado na aplicação.”
“O manual do processo foi de grande valia, uma vez que foi possível revisitar o processo sempre que possível para tirar dúvidas no decorrer do projeto.”
“O manual é bem completo, precisou ser checado algumas vezes durante o design da aplicação, e o sumário foi útil nessa questão, cobre bem todos os passos de requisitos e desenvolvimento. Por ser completo também é extenso, e poderia se beneficiar de um pequeno resumo dos passos no final do manual.”
“Devido a pouca experiência com documentação e desenvolvimento de aplicações voltadas a IOT, o processo MOTION foi de grande valia para que eu pudesse entender o que deveria ser feito e como realizar cada atividade, com ele foi possível a construção da documentação inicial do projeto sem muita dificuldade.”

Fonte: Próprio Autor

Tabela 24 – Comentários sobre a experiência de desenvolver as aplicações IoHT autoadaptativas propostas

Comentário
“A coleta de dados é um desafio principalmente no que se refere a dados de saúde. Esse foi o maior desafio presente no desenvolvimento para as aplicações em IoHT.”
“De lições aprendidas, a principal é a necessidade de entender bem o que o cliente necessita, desse modo, evita muito retrabalho. Como principal dificuldade foi garantir a segurança dos dados do usuário.”
“Vindo de um *background* de desenvolvimento web, as aplicações IoHT se apresentaram desafiadoras principalmente na parte visual, como apresentar dados sensíveis ao usuário sem parecer intrusivo, como eventos de noctúria, as ferramentas da IDE Android Studio para desenvolvimento de UIs também não são intuitivas. O que achei que seria mais difícil, pegar os dados dos sensores do celular, se tornou simples graças ao *framework* fornecido.”
“Foi bem interessante e desafiador, devido a pouca experiência com a área de IOT. Tive dificuldades para entender alguns termos e o funcionamento de tudo para documentar de forma clara. O uso do processo facilitou a execução das tarefas realizadas e pude levar um pouco da experiência que tive com a leitura e uso do processo, construir a documentação do segundo sistema sem o auxílio direto do mesmo.”

Fonte: Próprio Autor

6.3.3 Resultados das medidas para terceira hipótese

O resumo das respostas do questionário de *feedback* contendo as questões selecionadas relativas as duas medidas utilizadas para avaliar a terceira hipótese pode ser visto na Figura 40.

Para as respostas referentes a medida de percepção de Facilidade de aderência ao

processo MOTION (Figura 40a), na percepção de todos os profissionais de TIC, o processo MOTION se mostrou claro e compreensível (Q4). Além disso, todos concordaram que o processo MOTION é fácil de ser utilizado (Q5) e que pode facilmente guiar o desenvolvimento de aplicações (Q2). Para 87% dos profissionais de TIC foi fácil (Q6) e claro (Q3) usar o processo MOTION para desenvolver as aplicações, enquanto que um profissional de TIC se manteve neutro quanto à facilidade de uso geral do processo (Q6) e outro profissional de TIC se manteve neutro sobre a clareza (Q3) ao interagir com o processo MOTION durante o experimento. Por outro lado, 75% dos participantes do experimento concordam que foi fácil aprender a utilizar o processo MOTION (Q1) durante o experimento, enquanto dois participantes optaram pela neutralidade em relação a essa questão.

Figura 40 – Resultados das questões sobre percepção de facilidade de uso. Os valores Q1 a Q6 representam as questões do questionário

(a) Facilidade de aderência ao uso do processo MOTION

(b) Percepção de facilidade de desenvolvimento usando o processo MOTION

Fonte: Próprio autor

Em relação as respostas para a medida de Facilidade de desenvolvimento de sistemas IoHT autoadaptativos com o processo MOTION (Figura 40b), 87% dos profissionais de TIC concordam que aprender a usar o processo MOTION para desenvolver um sistema IoHT autoadaptativo foi fácil (Q1) e um participante escolheu a resposta neutra para essa questão. Os mesmos 87% dos participantes do experimento concordaram que é fácil utilizar o processo MOTION para guiar o desenvolvimento de um sistema IoHT autoadaptativo (Q2). Para todos os profissionais de TIC, a interação com o processo MOTION para desenvolver um sistema IoHT autoadaptativo foi clara e compreensível (Q3). Além disso, todos os profissionais de TIC concordaram que as atividades presentes no processo MOTION para desenvolvimento de um sistema IoHT autoadaptativo são claras e compreensíveis (Q4), que foi fácil se tornar hábil ao usar o processo MOTION para o desenvolvimento de um sistema IoHT autoadaptativo (Q5) e que o processo MOTION é fácil de usar para o desenvolvimento de um sistema IoHT autoadaptativo (Q6).

Tabela 25 – Medidas de da avaliação de facilidade de uso por profissional de TIC

Profissional de TIC	Média FAP	Média FAD
P1	5,00	5,00
P2	4,17	5,00
P3	4,17	4,50
P4	4,83	4,83
P5	4,00	4,00
P6	4,50	4,33
P7	4,50	4,17
P8	3,83	3,67

Fonte: Próprio Autor

Para análise estatística da terceira hipótese, calculados os valores médios das medidas de Facilidade de Aderência ao Processo (FAP) e Facilidade de uso do processo para desenvolvimento de um sistema IoHT autoadaptativo (FAD) para cada profissional de TIC (Tabela 25) e para cada questão (Tabela 26).

Tabela 26 – Medidas de da avaliação de facilidade de uso por questão

Questão	Média PUP	Média PUM
Q1	4,00	4,25
Q2	4,50	4,25
Q3	4,25	4,50
Q4	4,50	4,63
Q5	4,50	4,50
Q6	4,50	4,50

Fonte: Próprio Autor

Para o teste estatístico dessa hipótese, foram selecionados os testes t e de Wilcoxon, da mesma maneira que para a segunda hipótese, já que os dados coletados para a hipótese 3 seguem as mesmas características dos dados coletados para hipótese 2.

Para o teste t, as amostras foram verificadas em comparação ao valor 3, que é correspondente ao valor médio para a resposta das questões e referente ao valor de neutralidade na escala Likert e, para o teste de Wilcoxon, a comparação foi feita utilizando o valor 3 da mediana, do mesmo modo que foi feito para análise da segunda hipótese. Os resultados da análise das medidas para os dois testes podem ser vistos na Tabela 27.

É possível observar na Tabela 27 que tanto para o teste t quanto para o teste de Wilcoxon, em todos os cenários a significância foi inferior a 5%, que é correspondente ao valor crítico (p-value). Então, também é possível rejeitar a hipótese nula 3 em detrimento da hipótese alternativa 3.

Tabela 27 – Análise estatística das medidas de facilidade de uso

Medida	Referência	teste t	Wilcoxon
PUP	Valor por profissional de TIC	inferior à 0,01	0,006
PUM	Valor por profissional de TIC	inferior à 0,01	0,006
PUP	Valor por questão	inferior à 0,01	0,012
PUM	Valor por questão	inferior à 0,01	0,013

Fonte: Próprio Autor

6.4 Discussão

Para a hipótese 1, as medidas analisadas não permitem rejeitar a hipótese nula. Logo, não é possível afirmar que o uso do processo MOTION é capaz de melhorar a produtividade, a estabilidade de requisitos e a satisfação do time durante o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Ainda assim, pelas medidas coletadas há indícios de que pelo menos o uso do processo MOTION não parece afetar negativamente o desempenho do time ao desenvolver esses tipos de *softwares*. Além disso, embora não seja possível afirmar estatisticamente o impacto positivo do uso do processo na satisfação do time de desenvolvimento, é importante notar que a satisfação dos times foi maior ao desenvolver as aplicações quando utilizaram o processo MOTION. Além disso, vale destacar que o time 2, quando utilizou o processo MOTION, não pode contar plenamente com um de seus membros e ainda assim obteve resultados tão bons quanto o time 1 durante o desenvolvimento da aplicação 2 e até melhores nominalmente na maioria das medidas.

Seriam necessários mais experimentos para verificar melhor o impacto do uso do processo MOTION em relação ao desempenho do time durante o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Essa afirmação é reforçada por comentários de *feedback* recebidos durante as reuniões, onde os profissionais de TIC relataram ser vantajoso o uso do processo MOTION e que se sentiram mais seguros quando desenvolveram a aplicação junto com o processo.

Em relação às hipóteses 2 e 3, quanto à utilidade e facilidade de uso do processo, a análise dos dados das medidas coletadas para avaliação dessas hipóteses indicaram que foi possível rejeitar as hipóteses nulas em detrimento das alternativas e, com isso, percebe-se que os profissionais de TIC julgaram que o processo MOTION é útil e fácil de usar.

Além disso, pelos comentários fornecidos pelos profissionais de TIC no questionário e nos *feedbacks* das reuniões realizadas com esses profissionais durante o experimento, foi possível perceber que o processo MOTION auxiliou tanto aqueles que tinham mais conheci-

mento no desenvolvimento de aplicações dos domínios de IoHT e de sistemas autoadaptativos, quanto aqueles que tinham pouco ou nenhum conhecimento no desenvolvimento de aplicações especificamente para esses domínios.

Dados os resultados, é viável utilizar o processo MOTION em projetos de desenvolvimento para a criação de aplicações IoHT autoadaptativas baseadas em padrões de movimento, e, em oportunidades futuras, o processo MOTION pode ser utilizado por times de desenvolvimento em projetos diversos, tanto acadêmicos quanto na indústria para o desenvolvimento de soluções para IoHT.

Por fim, é importante destacar que o uso do *framework* KREATION em conjunto com o *template* de regras de adaptação ARTe teve impacto na produtividade dos times. Foi possível perceber que os profissionais de TIC participantes do experimento que possuíam pouca ou muita experiência no desenvolvimento de aplicações IoHT e autoadaptativas, conseguiram se aproveitar do *framework* para desenvolver mais rapidamente grande parte das funcionalidades das aplicações. Essa percepção foi reforçada nos *feedbacks* desses desenvolvedores durante as reuniões e nos comentários do questionário final. Isso também pode ser evidenciado em partes pelas medidas coletadas durante o experimento. Lembrando que a proposta principal do experimento era avaliar apenas o processo MOTION e seu manual de uso.

6.5 Ameaças à Validade

Esta Seção discute as ameaças à validade dos resultados do experimento com relação a: (i) Validade de Conclusão; (ii) Validade de Construto; (iii) Validade Interna; e (iv) Validade Externa.

As ameaças à Validade de Conclusão dizem respeito à relação entre o tratamento e o resultado (WHOLIN *et al.*, 2000; WOHLIN *et al.*, 2012). Entre essas ameaças está a confiabilidade das medidas utilizadas, o perfil do participantes do experimento, a divisão dos times, o baixo poder estatístico, a violação dos pressupostos dos testes estatísticos e os materiais utilizados.

Para obter medidas confiáveis, buscou-se na literatura medidas de avaliação de projetos de desenvolvimento consolidadas, que permitissem avaliar corretamente as hipóteses. Além disso, as medidas qualitativas foram baseadas no modelo TAM (SILVA, 2015; MARIKYAN; PAPAGIANNIDIS, 2022), que é utilizado para avaliar novas tecnologias, incluindo artefatos de *software* (ARAÚJO, 2022).

Com relação ao perfil dos participantes do experimento, foram escolhidos profissionais de TIC com um perfil mínimo que garantisse experiência com o desenvolvimento de *software*, incluindo desenvolvimento para dispositivos móveis, e que já tivessem utilizado processos de desenvolvimento de *software* anteriormente, incluindo métodos ágeis e iterativos, como o SCRUM. Já para divisão dos times, foi considerado um balanceamento no número de participantes de cada time e buscou-se garantir que ao menos um membro de cada time já tivesse experiência com aplicações de IoHT. No entanto, é importante destacar que foi considerado também a disponibilidade dos profissionais de TIC e, ainda assim, um dos membros de um dos times não conseguiu atuar focado durante todo o experimento, mas acredita-se que dados os valores de medidas coletadas, o impacto desse acontecimento não foi grande e afetou pouco os resultados.

Para garantir o poder estatístico da análise, foi feita uma verificação com relação ao tamanho da amostra e a forma como os dados coletados para as medidas foram distribuídos. Desse modo, chegou-se a conclusão que, dado o tamanho da amostra e a normalidade da distribuição dos dados verificada utilizando o teste de Kolmogorov-Smirnov (BERGER; ZHOU, 2014), fossem utilizados métodos não paramétricos para a análise da maioria das medidas. Ambos os métodos não paramétricos e paramétricos, foram utilizados para análise de algumas poucas medidas, quando a amostra, mesmo pequena, era significativa para essas medidas, dada a análise de outros estudos na literatura com o perfil de dados experimentais similares. O teste t (MARTINEZ; FERREIRA, 2007) (paramétrico) e o teste Wilcoxon (não paramétrico) (LAUREANO *et al.*, 2020; SIEGEL; JR, 2006) foram utilizados para avaliar amostras únicas ou relacionadas. Já para análise de amostras independentes, foi utilizado o teste não paramétrico de Mann-Whitney (CONTADOR; SENNE, 2016; CASTRO, 2012; SIEGEL; JR, 2006).

Em relação aos materiais, por ser um experimento onde os times desenvolveram as aplicações de maneira remota, onde cada profissional utilizou seu próprio *notebook*, não foi possível garantir uma configuração padrão das máquinas utilizadas, porém, nas reuniões contínuas com os profissionais, foi sempre verificado se havia restrições em relação ao equipamento, e os times em nenhum momento informaram restrições em relação a esse aspecto. Além disso, visando minimizar o impacto dessa ameaça, as medidas avaliadas durante o experimento não possuem relação ou são impactadas diretamente pela performance dos *notebooks* utilizados. Além disso, todos os dispositivos móveis usados para teste possuíam os sensores demandados pelas aplicações e todos os *wearables* utilizados para a segunda aplicação possuem a mesma

configuração e foram fornecidos pelo pesquisador.

A Validade de Construto diz respeito à generalização dos resultados do experimento, segundo (WHOLIN *et al.*, 2000; WOHLIN *et al.*, 2012). As principais ameaças estão relacionadas ao desenho do experimento, por exemplo, o viés da monooperação (WHOLIN *et al.*, 2000). Para mitigar essa ameaça, o experimento considerou o desenvolvimento de duas aplicações por dois times, onde um time funcionou como controle do outro. Além disso, o projeto e o planejamento do experimento foi previamente avaliado por especialistas, a fim de garantir que o experimento planejado era viável e capaz de retornar resultados confiáveis para o objeto da avaliação.

Segundo (SANTOS *et al.*, 2017; WHOLIN *et al.*, 2000), as ameaças à Validade Interna são influências que podem afetar as variáveis internas do experimento no que diz respeito à causalidade. Como uma dessas ameaças, pode-se destacar a escolha de um instrumento de avaliação ruim (SANTOS *et al.*, 2017; WOHLIN *et al.*, 2012). Porém, para mitigar essa ameaça, como já citado, todo o projeto e o planejamento do experimento foi previamente avaliado por especialistas na avaliação de diversos domínios de aplicação, incluindo especialistas em aplicações de IoHT. Os *feedbacks* dos especialistas permitiram avaliar e melhorar o planejamento do experimento.

Finalmente, de acordo com (SANTOS *et al.*, 2017; WHOLIN *et al.*, 2000), as ameaças à Validade Externa correspondem a condições que limitam a capacidade de generalizar os resultados. A principal ameaça a essa validade está nas aplicações escolhidas e no status esperado das mesmas ao fim de cada parte do experimento. As aplicações escolhidas utilizam poucos sensores e possuem poucas regras de adaptação, além disso, não era esperado que todos os artefatos possíveis para cada aplicação fossem gerados e que as versões finais apresentadas para cada aplicação contivesse todas funcionalidades implementadas. Porém, para mitigar essa questão, foi solicitado aos times que um conjunto mínimo de artefatos fossem entregues para cada aplicação desenvolvida, necessitando que os participantes de cada time se dedicassem a várias etapas do processo de desenvolvimento de *software*. Um experimento com aplicações IoHT mais complexas seria incompatível com o tempo disponível, incluindo o tempo que os profissionais de TIC podiam se dedicar ao experimento.

6.6 Conclusão

Este capítulo apresentou a avaliação do processo MOTION através de um experimento com profissionais de TIC. Esse experimento permitiu avaliar, com base na percepção dos participantes do experimento, a utilidade e facilidade de uso do processo MOTION.

No próximo capítulo a hipótese de pesquisa é revisitada e são apresentadas as conclusões e trabalhos futuros desta tese de doutorado.

7 CONCLUSÃO

Esse trabalho propôs um processo de desenvolvimento de *software*, intitulado MOTION, para guiar o desenvolvimento de aplicações de *Internet of Health Things* autoadaptativas baseadas em padrões de movimento. Além disso, também foram propostos três artefatos de reúso de *software* para auxiliar o desenvolvimento de aplicações IoHT autoadaptativas, que podem ser utilizados em conjunto com o processo MOTION.

Este capítulo conclui esta tese e está dividido da seguinte maneira: na Seção 7.1 é apresentada a visão geral do trabalho; na Seção 7.2 são resumidos os principais resultados alcançados; as hipóteses da pesquisa são revisitadas e é feita a comparação com os trabalhos relacionados na Seção 7.3; as limitações do trabalho são descritas na Seção 7.4; e, por fim, na Seção 7.5 são apresentados as propostas de trabalhos futuros.

7.1 Visão Geral

Informações sobre o movimento, como a marcha, a velocidade, a postura e a localização podem ser utilizados para identificar problemas de saúde, principalmente em idosos (KIRKWOOD *et al.*, 2018). Para coletar esses dados e identificar de maneira automatizada padrões de movimento, é possível utilizar dispositivos inteligentes e soluções de Internet das Coisas (do inglês, *Internet of Things* - IoT). A IoT propõe uma infraestrutura onde dispositivos dotados de sensores e de algum tipo processamento se integram através da internet, que é uma rede global, dinâmica, autoconfigurável e interoperável (SUNDMAEKER *et al.*, 2020).

O uso de IoT para saúde vem sendo chamado de *Internet of Health Things* (IoHT) (RODRIGUES *et al.*, 2018). Muitas das soluções recentes de IoHT utilizam diferentes padrões de movimento para identificar problemas de saúde, como doenças crônicas ou situações de risco à saúde, como quedas (PEIMANKAR *et al.*, 2023; LINHARES *et al.*, 2020; TARAMASCO *et al.*, 2018). No entanto, ainda existem questões em aberto a serem tratadas por esse campo de pesquisa. Entre essas questões estão a ausência de um processo de desenvolvimento de aplicações IoHT baseadas em padrões de movimento (SYED *et al.*, 2019), a falta de soluções sobre como relacionar os dados coletados pelos sensores dos dispositivos IoHT (SUN *et al.*, 2020; MAJUMDER *et al.*, 2019) e as diferentes situações de saúde, e a necessidade de criar soluções mais ubíquas (BRAVO *et al.*, 2018).

Na revisão de literatura do Capítulo 3, observou-se que a grande maioria das propos-

tas de como relacionar dados de movimento coletados por sensores e situações de saúde eram específicas. Alguns trabalhos recentes propunham modelos para reúso em diferentes soluções (DÍAZ *et al.*, 2018; AICHA *et al.*, 2018; LAZAROU *et al.*, 2016), mas ainda com espaço para proposição de outros artefatos de reúso, que permitissem também auxiliar a tomada de decisão sobre as escolhas de quais sensores utilizar para identificar situações de saúde específicas. Já em relação a construção de aplicações mais ubíquas, a solução inovadora apresentada nesta tese foi o uso de sistemas IoHT autoadaptativos, capazes de se adaptar automaticamente em resposta a mudanças no seu contexto (BRUN *et al.*, 2009).

Considerando tudo isso, essa tese de doutorado propôs um processo de desenvolvimento de software para aplicações IoHT autoadaptativas baseadas em padrões de movimento, intitulado MOTION, e um conjunto de artefatos de reúso de software para auxiliar o desenvolvimentos de aplicações IoHT, que podem ser utilizados em conjunto com o processo proposto. Esses artefatos de reúso consistem em: (i) um modelo de grafo de classificação, intitulado GRAFIT, que relaciona sensores, *features*, algoritmos de classificação e situações ou estados de saúde; (ii) um *template* para definição de regras de adaptação, intitulado ARTe; e (iii) um *framework* em Kotlin, chamado KREATION, para o desenvolvimento de aplicações IoHT autoadaptativas para dispositivos móveis.

O MOTION é baseado no processo orientado a reúso de componentes (SOMMERVILLE, 2010), sendo adicionadas atividades específicas para a construção de sistemas IoHT com padrões de movimento e de sistemas autoadaptativos. Já os artefatos de reúso de software propostos foram construídos com base em um conjunto de atividades e necessidades comuns a diferentes aplicações IoHT baseadas em padrões de movimento, identificadas durante um mapeamento sistemático da literatura. O modelo GRAFIT relaciona sensores, *features* de movimento, algoritmos inteligentes para classificação e situações de saúde. Esse artefato pode auxiliar a elicitação de requisitos e servir como base de conhecimento para a aplicação. Já o *template* ARTe auxilia a construção de regras de adaptação para aplicações autoadaptativas e pode ser utilizado para auxiliar tanto a implementação quanto o projeto da aplicação. Por último, o *framework* KREATION combina vários componentes comuns a aplicações IoHT, com módulos que incorporam o *loop* de adaptação dos sistemas autoadaptativos. Além disso, o *framework* utiliza diferentes padrões de software, como os padrões *observer*, para a comunicação entre os módulos, e DAO, para o gerenciamento de bancos de dados. O KREATION também incorpora o *framework* SUCCEEd (JUNIOR *et al.*, 2018), que auxilia na orquestração das ações em sistemas

autoadaptativos. O KREATION possui também facilidades para auxiliar o uso do GRAFIT e do *template* ARTe propostos nessa tese. Tanto o processo MOTION quanto os demais artefatos foram apresentados no Capítulo 4 desta tese.

No Capítulo 5 foram apresentadas as avaliações do processo MOTION e dos artefatos de reúso através de duas provas de conceito. A primeira prova de conceito foi utilizada para avaliar a versão inicial do processo MOTION e do modelo GRAFIT. Uma segunda PoC, também utilizando o MOTION e contemplando o desenvolvimento de duas aplicações móveis, foi utilizada para a avaliar o *framework* KREATION e o *template* ARTe.

No capítulo 6 foi apresentado um experimento com profissionais de Tecnologia da Informação e Comunicação (TIC), que utilizaram o processo MOTION para desenvolver duas aplicações móveis ao longo de seis semanas. Com as PoCs foi possível avaliar e demonstrar a viabilidade de uso e funcionalidade de todos os artefatos de reúso de software propostos nessa tese e do processo MOTION para auxiliar a construção de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Já os resultados dos experimentos mostraram que, na perspectiva de profissionais de TIC, o processo MOTION é útil e fácil de utilizar para o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento.

7.2 Principais Resultados

Os principais resultados desta tese, que foram apresentados no Capítulo 4 e avaliadas nos Capítulos 5 e 6, são resumidos a seguir:

- **Processo MOTION:** o principal resultado dessa tese foi o processo de desenvolvimento de *software* MOTION para auxiliar o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Esse processo adapta o modelo de processo de desenvolvimento de *software* orientado à reúso de componentes e adiciona atividades a esse processo, a fim de contemplar características dos sistemas autoadaptativos e de aplicações IoHT baseados em padrões de movimento. Vale ressaltar que o processo MOTION considera tanto o desenvolvimento do *software* em tempo de projeto, quanto a autoadaptação do sistema em tempo de execução.
- **Modelo GRAFIT:** o GRAFIT é um artefato que modela em forma de grafo o relacionamento entre sensores, *features*, algoritmos de classificação e situações ou estados de saúde. Esse grafo pode auxiliar etapas diversas do processo de desenvolvimento de *software*, como a elicitação de requisitos e projeto da aplicação, bem como pode servir como base

de conhecimento para ser utilizada em conjunto pela aplicação implementada.

- ***Template* ARTe:** o *template* ARTe provê um padrão para criar regras de adaptações utilizando *tags* em um arquivo ".xml", que pode ser reutilizado por aplicações autoadaptativas.
- ***Framework* KREATION:** o KREATION é um *framework* desenvolvido usando a linguagem Kotlin e auxilia a implementação de aplicações IoHT autoadaptativas para dispositivos Android. Esse *framework* incorpora o *loop* de adaptação MAPE-K, através de módulos que se comunicam utilizando o padrão *observer*. O *framework* KREATION segue a arquitetura MVC e utiliza uma lógica de coleta de dados de sensores do dispositivo Android similar à proposta pelo *midleware* LoCCAM (MAIA *et al.*, 2013). Além disso, ele incorpora o *framework* SUCCEEd para auxiliar a orquestração das ações do sistema autoadaptativo e possui facilidades para uso de instâncias do modelo GRAFIT e de um arquivo de regras de adaptação baseado no *template* ARTe, propostos nessa tese.

Durante o desenvolvimento desta tese também foram obtidos resultados secundários que contribuíram para a sua construção e que são descritos a seguir:

- **Aplicações REALER e ARCANA:** desenvolvimento das aplicações REALER e ARCANA, utilizando o *framework* KREATION, as quais permitem exemplificar a implementação de diferentes aplicações com esse *framework* em conjunto com o processo MOTION. O código da aplicação REALER está disponível em: https://bit.ly/3N6viCW. Já o código da aplicação ARCANA está disponível em: .https://bit.ly/42INfxk.
- **Implementação do GRAFIT:** uma versão do grafo de classificação foi implementada, utilizando duas bases de dados de movimentos, com dados coletados de sensores de acelerômetro e giroscópio, e pode ser reusado por outras aplicações. O código do grafo de classificação implementado e as bases de dados estão disponíveis em: https://bit.ly/3p0xO5P.
- **Exemplos de arquivos de regras de adaptação baseadas no *template* ARTe:** exemplos de arquivos de regras de adaptação, baseada no *template* ARTe e no contexto de bateria de dispositivos móveis, foram construídos para uso em conjunto com as aplicações REALER e ARCANA. Esses arquivos e um exemplo de código de uma API Web para disponibilização dos mesmos está disponível em: https://bit.ly/465yY0q.

Tabela 28 – Artigos na área de IoHT desenvolvidos durante o doutorado

Referência	Qualis	Status
COSTA JUNIOR, E.; ANDRADE, R.M.C.; ROCHA, L. S. *KREATION: Kotlin Framework for Self-adaptive IoHT Applications*. 11th IEEE International Conference on Serious Games and Applications for Health, 2023.	A3	Aceito
COSTA JUNIOR, E.; ANDRADE, R.M.C.; ROCHA, L. S. *Classification Graph to the Internet of Health Things Applications*. 11th IEEE International Conference on Healthcare Informatics, 2023.	A3	Aceito
COSTA JUNIOR, E.; ANDRADE, R.M.C.; SANTOS, I.S.; OLIVEIRA, P.A.M.; VENCESLAU, A.D.; OLIVEIRA, B. S. *Where Is the Internet of Health Things Data?*. 24th International Conference on Enterprise Information Systems, 2022.	A3	Publicado
OLIVEIRA, P.A.M.; ANDRADE, R.M.C.; **COSTA JUNIOR, E.**; SANTOS, I.S. *Ten Years of eHealth Discussions on Stack Overflow*. International Conference on Health Informatics, 2022.	A3	Publicado
COSTA JUNIOR, E.; ANDRADE, R. M. C.; ROCHA, L. S.; *Development Process for Self-adaptive Applications of the Internet of Health Things based on Movement Patterns*. 9th IEEE International Conference on Healthcare Informatics (Doctoral Consortium), 2021.	-	Publicado
COSTA JUNIOR, E.; ANDRADE, R. M. C.; ROCHA, L. S.; TARAMASCO, C.; FERREIRA, L. *Computational Solutions for Human Falls Classification*. IEEE Access, 2021.	A1	Publicado
OLIVEIRA, B. S.; ARAÚJO, Í. L.; PAIVA, J. O. V.; **COSTA JUNIOR, E.**; ANDRADE, R. M. C. *Refactoring Decision based on Measurements for IoHT Apps*. XVII Brazilian Symposium on Information Systems, 2021, Uberlândia Brazil.	A4	Publicado
ARAUJO, I. L.; ANDRADE, R. M. C.; **COSTA JUNIOR, E.**; DUARTE, P. A. S.; SANTOS, I. S.; OLIVEIRA, P. A. M. *Towards a Taxonomy for the Development of Older Adults Healthcare Applications*. 53rd Hawaii International Conference on System Sciences, 2020, Maui - Hawaii.	A1	Publicado
OLIVEIRA, P. A.; ARAUJO, I. L.; **COSTA JUNIOR, E.**; DUARTE, P. A. S.; SANTOS, I. S.; ANDRADE, R. M. C.; BARRETO, I. C. H. C.; ANDRADE, L. O. M. *Dorsal: Ferramenta para Geração de Modelos de Dados para Aplicações voltadas a Saúde e Cuidado de Idosos*. 20º Simpósio Brasileiro de Computação Aplicada à Saúde, 2020, Salvador (BA).	A4	Publicado
ARAUJO, I. L.; ANDRADE, R. M. C.; **COSTA JUNIOR, E.**; OLIVEIRA, P. A. M.; OLIVEIRA, B.; AGUILAR, P. *Lessons Learned from the Development of Mobile Applications for Fall Detection*. The Ninth International Conference on Global Health Challenges, 2020.	-	Publicado

Fonte: Próprio Autor

Durante o desenvolvimento desta pesquisa de doutorado, também foram publicados 13 artigos, sendo dez (10) relacionados à tese e a área de IoHT e três (3) relacionados a outros temas que contribuíram para o amadurecimento do aluno como pesquisador. Esses artigos estão elencados nas Tabelas 28 e 29, respectivamente.

Tabela 29 – Outros artigos desenvolvidos durante o doutorado

Referência	Qualis	Status
COSTA JUNIOR, E.; COSTA, W. L.; PORTELA, A. L. C. ; ROCHA, L. S.; GOMES, R.; ANDRADE, R. M. C. *Detecting Attacks and Locating Malicious Devices Using Unmanned Air Vehicles and Machine Learning*. JOURNAL OF INTERNET SERVICES AND APPLICATIONS, 2022.	A1	Publicado
COSTA JUNIOR, E.; GOMES, R.; ROCHA, L. S.; ANDRADE, R. M. C. *Localização de Dispositivos Maliciosos usando Veículos Aéreos não Tripulados*. Simpósio Brasileiro de Computação Ubíqua e Pervasiva, 2020, Brasil. p. 141.	B4	Publicado
SANTOS, I. S.; **COSTA JUNIOR, E.**; ANDRADE, R. M. C. ; SANTOS NETO, P. A.; SAMPAIO, L.; WERNER, C. M. L. ; SOUZA, J. T. *Optimized Feature Selection for Initial Launch in Dynamic Software Product Lines*. 20th International Conference on Enterprise Information Systems, 2018.	A3	Publicado

Fonte: Próprio Autor

Durante o período do doutorado, o autor também participou de projetos de pesquisa, desenvolvimento e inovação relacionados ao tema de saúde. Entre 2019 e 2020, o autor participou do "Programa Fiocruz de Fomento a Inovação: Inteligência de Governança para Tomada de Decisão na Gestão de Sistemas de Saúde com Enfoque no Cuidado em Saúde do Idoso". Entre 2020 e 2022, ele participou de dois projetos internacionais do programa Stic/Amsud em parceria com instituições francesas e de vários países da América do Sul, dentre os quais Brasil, Chile, Colômbia, Peru e Uruguai. Esses projetos do programa Stic/Amsud foram: "Rise e-Well: *Remote Intelligent Systems for Elderly and Chronic Patient Follow-up*" e "Angel: *IoT e-Health Platform to Monitor and Improve Quality of Life*".

Além disso, o autor dessa tese coorientou alunos de iniciação científica participantes dos projetos Pibic relacionados à área de saúde digital intitulados "GREat IoT - Internet das Coisas aplicada à Saúde" (entre 2020 e 2021) e "Angel: IoT e-Health Platform para monitorar e melhorar a qualidade de vida" (entre 2021 e 2022).

7.3 Revisitando Hipóteses e Trabalhos Relacionados

Com base nos resultados apresentados no Capítulo 4 e nas avaliações descritas no Capítulo 5, é possível analisar (aceitar ou rejeitar) a hipótese de pesquisa desta tese apresentada no Capítulo 1. Ambas, hipótese e análise, são descritas a seguir.

Hipótese de Pesquisa
No desenvolvimento de aplicações IoHT baseadas em padrões de movimento, o uso de um processo de desenvolvimento de *software* com o conceito de autoadaptação pode ser útil para facilitar o desenvolvimento dessas aplicações e prover uma maneira viável de relacionar dados de sensores e situações de saúde.

Análise da Hipótese: **Aceita**. Baseada nas avaliações executadas foi possível demonstrar a viabilidade, utilidade e facilidade de uso do processo MOTION, o qual auxilia o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento, com o auxílio dos 3 (três) artefatos de reúso de *software* propostos neste trabalho.

Também foi demonstrado que, na percepção de profissionais de TIC, o processo MOTION e seu manual de execução são fáceis de usar e podem ser utilizados por novas equipes de desenvolvimento sem exigir uma grande curva de aprendizagem.

Com relação aos trabalhos relacionados apresentados no Capítulo 3, embora não tenha sido encontrado na literatura nenhum outro processo de desenvolvimento de *software* específico para aplicações de IoHT, foram identificados alguns métodos que guiaram o desenvolvimento de aplicações IoHT baseadas em padrões de movimento específicas, como aplicações para reconhecimento de atividades do dia a dia e fragilidades (SUN *et al.*, 2020)(JALAL *et al.*, 2019)(KAMPEL *et al.*, 2018), identificação de movimentos anômalos (DÍAZ *et al.*, 2018) e detecção de questões de saúde com base na trajetória do movimento (AICHA *et al.*, 2018). Em contraponto a esses métodos, o processo MOTION é mais generalista, podendo ser utilizado para o desenvolvimento de diferentes aplicações de IoHT baseadas em padrões de movimento, incluindo as cobertas por esses métodos. E, por suportar o desenvolvimento de sistemas IoHT autoadaptativos, ele permite desenvolver sistemas ubíquos, adaptados às mudanças contínuas no ambiente em que estão inseridas as aplicações IoHT. Além disso, diferente dos métodos apresentados, o processo MOTION não define um conjunto de sensores (SUN *et al.*, 2020; JALAL *et al.*, 2019; AICHA *et al.*, 2018; KAMPEL *et al.*, 2018) e nem uma estrutura de dados (AICHA *et al.*, 2018) específicos, mas pode auxiliar o desenvolvimento de aplicações IoHT baseadas em padrões de movimento com diversos tipos de sensores e de estruturas de dados. Um

resumo dessa análise pode ser visto na Tabela 30.

Tabela 30 – Processo MOTION em comparação com os métodos dos trabalhos relacionados

Trabalho	Sensores	Objetivo	Estrutura de dados	Avaliação	Adaptado para SAS
Processo MOTION	Sensores diversos em dispositivos IoHT	Auxiliar o desenvolviemnto de aplicações IoHT autoadaptativas baseadas em padrões de movimento	Pemite diversas	PoCs e Experimento	Sim

Fonte: Próprio Autor

Nos trabalhos relacionados também foram apresentadas pesquisas com propostas de modelos para auxiliar a implementação de aplicações IoHT baseados em padrões de movimento (KHAN *et al.*, 2022) (KODYŠ *et al.*, 2017) (BAIG *et al.*, 2016). Em geral, são modelos de dados que podem ser utilizados para modelar situações de saúde específicas, que são inferidas com base em padrões de movimento. Nessa tese é proposto um modelo de grafo de classificação, chamado GRAFIT, para relacionar sensores, *features*, algoritmos de classificação e situações de saúde. O modelo de grafo pode tanto ser utilizado para auxiliar a tomada de decisão em partes do desenvolvimento das aplicações IoHT baseadas em padrões de movimento, como na elicitação de requisitos, por exemplo, provendo uma lista de quais possíveis sensores podem ser utilizados para identificar uma situação de saúde desejada. Um grafo de classificação também pode ser instanciado e ser reaproveitado como base de conhecimento para uma aplicação IoHT.

De maneira similar a (KHAN *et al.*, 2022), além de apresentar um grafo para a modelagem, o artefato de reúso proposto nessa tese foi projetado para processar os dados apenas em um servidor, que pode estar em nuvem, embora, como já citado, a estrutura de grafo após criada (ou instanciada) pode ser baixada e utilizada pela aplicação móvel. Enquanto em (KHAN *et al.*, 2022), (KODYŠ *et al.*, 2017) e (BAIG *et al.*, 2016) os modelos foram avaliados com experimentos, o modelo proposto por essa tese foi avaliado com base em uma PoC. A avaliação do grafo de classificação utilizando um experimento é um possível trabalho futuro.

Um resumo da análise comparativa do modelo GRAFIT em relação aos modelos apresentados nos trabalhos relacionado da Seção 3.3 do Capítulo 3 pode ser visto na Tabela 31.

Por fim, três trabalhos relacionados apresentaram *frameworks* para o desenvolvimento de soluções IoHT (SYED *et al.*, 2019) (BALAJI *et al.*, 2018) (MUKHERJEE *et al.*, 2021). Do mesmo modo que nesses estudos, é proposto nessa tese um *framework*, intitulado KREATION, para o desenvolvimento de aplicações IoHT. No entanto, o KREATION foi projetado

Tabela 31 – Modelo GRAFIT em comparação com os modelos dos trabalhos relacionados

Trabalho	Sensores	Objetivo	Estrutura de dados	Local do processamento	Avaliação
Modelo GRAFIT	Sensores diversos em dispositivos IoHT	Modelar o relacionamento entre sensores, *feature*, algoritmos de classificação e situações de saúde	Grafo	Na nuvem	PoC

Fonte: Próprio Autor

para auxiliar o desenvolvimento de aplicações IoHT autoadaptativas para dispositivos Android, enquanto que os *frameworks* apresentados nos trabalhos relacionados são mais generalistas e não consideram sistemas autoadaptativos. Além disso, os *frameworks*, por serem mais generalistas, focam na estrutura de um sistema IoHT para ambientes inteligentes, contando com a aplicação nos dispositivos e o processamento em nuvem dos dados e não fornecem bibliotecas, módulo e estruturas específicas para o desenvolvimento das aplicações IoHT móvel. Por outro lado, o *framework* KREATION tem como foco a aplicação móvel e não fornece estruturas para processamento em nuvem dos dados.

Em relação ao tipo de *hardware*, os *frameworks* propostos por (SYED *et al.*, 2019) e (MUKHERJEE *et al.*, 2021) utilizam dispositivos móveis e *wearables*, do mesmo jeito que o *framework* KREATION, enquanto que o *framework* proposto por (BALAJI *et al.*, 2018) trabalha tanto com *wearables*, quanto com outros dispositivos IoT dispostos no ambiente. Já o processamento de dados de todos os *frameworks* propostos nos trabalhos relacionados é feito em um servidor de nuvem, embora, os *frameworks* de (BALAJI *et al.*, 2018) e (MUKHERJEE *et al.*, 2021) também sejam adaptados para processar dados na borda da rede ou na névoa. Já o KREATION processa apenas dados locais, porém, é possível, usando o KREATION, utilizar estruturas de dados que são processadas em nuvem e depois baixadas para serem reutilizadas no dispositivo, como no caso do grafo de classificação proposto nessa tese. Por fim, enquanto que os *frameworks* propostos nos trabalhos relacionados trabalham tanto com aplicações Web, quanto com aplicações móveis, o KREATION foca nas aplicações móveis para Android. Um resumo da comparação entre o *framework* KREATION e os demais *frameworks* da Seção 3.4 do Capítulo 3 pode ser visto na Tabela 32 abaixo.

Tabela 32 – *Framework* KREATION em comparação com os *frameworks* dos trabalhos relacionados

Trabalho	Tipos de Dispositivos IoT	Armazenamento e Tratamento de dados	Processamento dos dados	Visualização de dados	Adaptado para SAS
***Framework* KREATION**	Dispositivos Android	Armazenamento e Manipulação de dados coletados dos sensores do dispositivo	No dispositivo	Apenas Mobile	Sim

Fonte: Próprio Autor

7.4 Limitações

O processo MOTION e os demais artefatos de reúso propostos nessa tese auxiliam o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. No entanto, estes resultados apresentam algumas limitações, detalhadas a seguir.

O processo MOTION, na versão desta tese, só foi avaliado para o desenvolvimento de aplicações móveis, o que compreende uma grande parte dos sistemas IoHT, porém, existem sistemas IoHT para ambientes *Desktop* e Web e seria necessário mais testes para avaliar o uso do MOTION para sistemas desse tipo. Além disso, o experimento que avaliou o processo considerou times pequenos de profissionais de TIC por um pequeno período de tempo. Seria necessário mais experimentos com times maiores e por um período maior, para avaliar a utilidade do processo e se há melhora no desempenho dos times com o uso do processo MOTION nessas circunstâncias.

Em relação ao modelo GRAFIT, há uma limitação em relação a proposição de seu processo de atualização, que é feito de maneira "*offline*" na versão desta tese, de modo que é necessário sempre iniciar a atualização do grafo com base nas bases de dados fornecidas. No entanto, o grafo de classificação pode evoluir para também ser atualizado de maneira automática e "online" considerando também os dados coletados pelas aplicações móveis que o utilizam como base de conhecimento, de maneira que esses dados possam ser enviados a um servidor e possam ser acessados pelo sistema de gerenciamento do grafo para retreinar os modelos dos algoritmos de classificação e atualizar o grafo de classificação em paralelo a execução da aplicação.

Já em relação ao *framework* KREATION, sua principal limitação está no fato de ser desenvolvido em Kotlin, de modo que é voltado apenas para o desenvolvimento de aplicações Android nativas. Essa escolha foi feita pela flexibilidade e facilidade de uso da plataforma Android e pela melhor qualidade e facilidade de manipulação dos sensores dos dispositivos Android por aplicações nativas.

Por último, os artefatos de reúso foram avaliados utilizando apenas PoCs, embora

tenham sido testados em parte pelos profissionais de TIC durante o experimento para avaliação do processo MOTION. Seria interessante em pesquisas futuras desenvolver também outros experimentos com casos reais voltados a avaliação dos artefatos de reúso propostos.

7.5 Trabalhos Futuros

Esta tese propôs o processo MOTION e três artefatos de reúso de *software*, que visam auxiliar o desenvolvimento de aplicações IoHT autoadaptativas baseadas em padrões de movimento. Os principais trabalhos de pesquisa futuros derivados dos resultados desta tese são descritos a seguir:

- **Refinamento do processo MOTION para as etapas de Validação e Verificação em tempo de execução**. Embora o processo MOTION apresente atividades para validação e verificação tanto em tempo de projeto quanto em tempo de execução, seria interessante aprofundar um pouco mais a etapa de validação e verificação em tempo de execução, com o objetivo de incorporar ao manual de execução do processo, os métodos recentes dessa área de pesquisa. Essa área de pesquisa vem ganhando mais atenção nos últimos anos e ainda possui muitos desafios em abertos a serem explorados, como mostrado em (SANTOS *et al.*, 2017).
- **Evolução do GRAFIT**. Como mencionado nas limitações, seria interessante em trabalhos futuros, a proposição de um processo para atualização "online" dos conjuntos de dados usados pelo GRAFIT, no qual o grafo de classificação, implementado com base no modelo GRAFIT, pode ser atualizado com base nos dados coletados pela aplicação IoHT móvel e enviados para um servidor em nuvem. Esse processo poderia ser inclusive pensado para funcionar como uma possível autoadaptação da aplicação. Além disso, seria interessante verificar se outros formatos de arquivo, como ".json", seriam mais vantajosos para visualização e reutilização do grafo de classificação do que o formato ".xml" utilizado na primeira PoC dessa pesquisa. Seria interessante também fazer análises comparativas da acurácia e precisão de algoritmos de classificação utilizados em versões implementadas do GRAFIT, como os usados na primeira PoC, com o objetivo de identificar se o uso do GRAFIT impacta de algum modo na maneira como esses algoritmos são implementados e utilizados em aplicações de IoHT.
- **Ferramenta de visualização do grafo de classificação**. Além da visualização do grafo de classificação implementado utilizando um arquivo, em trabalhos futuros, seria interessante

desenvolver uma ferramenta para auxiliar a visualização do grafo de classificação tanto em dispositivos móveis quanto em um ambiente Web. Essa ferramenta pode ajudar os times de desenvolvimento que desejam utilizar um grafo classificação como suporte à tomada de decisão em etapas de desenvolvimento do software, como a elicitação de requisitos ou o projeto da aplicação.

- **Evolução do *template* ARTe**. Na versão desta tese do *template* ARTe é utilizado um arquivo ".xml" para definição das regras de adaptação. Porém, seria interessante verificar outras abordagens para definição de regras de adaptação também, como, por exemplo, o uso de programação orientada a aspectos, como utilizado para testes de SAS em tempo de execução no trabalho de (SANTOS, 2020).
- **Evolução do *framework* KREATION**. Várias possíveis evoluções no *framework* KREATION poderiam ser analisadas em trabalhos futuros, dentre as quais destaca-se: (i) a incorporação de um módulo de segurança, que não apenas cuide da privacidade e anonimização dos dados, mas também garanta a confiabilidade dos dados coletados dos dispositivos móveis, ou ainda trafegados para algum servidor em nuvem, como é comum para aplicações de IoHT; (ii) a possibilidade de adaptar o *framework* para auxiliar também o desenvolvimento de serviços e/ou microsserviços para aplicações que utilizam arquiteturas orientadas a serviços; e (iii) gerar versões do *framework* para outras linguagens de programação, por exemplo, Swift, a fim de permitir o uso da estrutura do *framework* KREATION para auxiliar a implementação de aplicações para outras plataformas diferentes do Android.
- **Mais avaliações empíricas**. Seria interessante o desenvolvimento de mais experimentos utilizando tanto o processo MOTION quanto os artefatos de reúso para analisar outros aspectos do processo e dos artefatos, que não foram possíveis avaliar durante o tempo de pesquisa dessa tese. Nessas novas análises empíricas, seria interessante avaliar melhor o impacto do uso também dos artefatos de softwares propostos nessa tese em relação ao desempenho e velocidade do time.
- **Uso do processo MOTION em projetos reais**. Sendo o MOTION um processo de desenvolvimento de *software* para sistemas IoHT autoadaptativos baseados em padrões de movimento, seria importante acompanhar e analisar como o uso desse processo pode impactar o desenvolvimento de aplicações IoHT reais, inclusive em aplicações que utilizam dispositivos IoT diferentes dos usados nas PoCs e no experimento apresentado nessa tese.

REFERÊNCIAS

AICHA, A. N.; ENGLEBIENNE, G.; KRÖSE, B. Continuous measuring of the indoor walking speed of older adults living alone. **Journal of ambient intelligence and humanized computing**, Springer, v. 9, n. 3, p. 589–599, 2018.

AJERLA, D.; MAHFUZ, S.; ZULKERNINE, F.; PRYSS, R. C. A real-time patient monitoring framework for fall detection. **Wireless Communications and Mobile Computing**, John Wiley and Sons Ltd., v. 2019, p. 13, jan 2019. ISSN 1530-8669.

AL-FUQAHA, A.; GUIZANI, M.; MOHAMMADI, M.; ALEDHARI, M.; AYYASH, M. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE communications surveys & tutorials**, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

ANDERSSON, J.; BARESI, L.; BENCOMO, N.; LEMOS, R. D.; GORLA, A.; INVERARDI, P.; VOGEL, T. Software engineering processes for self-adaptive systems. In: **Software Engineering for Self-Adaptive Systems II**. [*S. l.*]: Springer, 2013. p. 51–75.

ARAÚJO, Í. L. d. **Ágape: Framework para detecção de quedas de idosos e suas causas com dispositivos vestíveis**. Tese (Doutorado) – Universidade Federal do Ceará, 2022.

ARCHAMBAULT, É.; CAMPBELL, D.; GINGRAS, Y.; LARIVIÈRE, V. Comparing bibliometric statistics obtained from the web of science and scopus. **Journal of the American society for information science and technology**, Wiley Online Library, v. 60, n. 7, p. 1320–1326, 2009.

ASIF, F. **Ultra-fast Machine Learning Inference for triggering at CMS Experiment**. [*S. l.*], 2021.

BAIG, M. M.; GHOLAMHOSSEINI, H.; CONNOLLY, M. J. Falls risk assessment for hospitalised older adults: a combination of motion data and vital signs. **Aging clinical and experimental research**, Springer, v. 28, n. 6, p. 1159–1168, 2016.

BALAJI, R.; BHAVSAR, K.; BHOWMICK, B.; MITHUN, B.; CHAKRAVARTY, K.; CHATTERJEE, D.; GHOSE, A.; GUPTA, P.; JAISWAL, D.; KIMBAHUNE, S. *et al.* A framework for pervasive and ubiquitous geriatric monitoring. In: SPRINGER. **International Conference on Human Aspects of IT for the Aged Population**. [*S. l.*], 2018. p. 205–230.

BARBOSA, D. M.; LIMA, R. G. D. M.; MAIA, P. H. M.; COSTA, E. Lotus@ runtime: a tool for runtime monitoring and verification of self-adaptive systems. In: IEEE. **2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. [*S. l.*], 2017. p. 24–30.

BARRY, L. C.; HATCHMAN, L.; FAN, Z.; GURALNIK, J. M.; GAO, R. X.; KUCHEL, G. A. Design and validation of a radio-frequency identification-based device for routinely assessing gait speed in a geriatrics clinic. **Journal of the American Geriatrics Society**, Wiley Online Library, v. 66, n. 5, p. 982–986, 2018.

BELBACHIR, A.; DROBICS, M.; MARSCHITZ, W. Ambient assisted living for ageing well – an overview. **e & i Elektrotechnik und Informationstechnik**, v. 127, p. 200–205, 08 2010.

BERGER, V. W.; ZHOU, Y. Kolmogorov–smirnov test: Overview. **Wiley statsref: Statistics reference online**, Wiley Online Library, 2014.

BRAVO, J.; HERVÁS, R.; FONTECHA, J.; GONZÁLEZ, I. M-health: lessons learned by m-experiences. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 18, n. 5, p. 1569, 2018.

BRUN, Y.; SERUGENDO, G. D. M.; GACEK, C.; GIESE, H.; KIENLE, H.; LITOIU, M.; MÜLLER, H.; PEZZÈ, M.; SHAW, M. Engineering self-adaptive systems through feedback loops. In: **Software engineering for self-adaptive systems**. [*S. l.*]: Springer, 2009. p. 48–70.

BRUNO, B.; MASTROGIOVANNI, F.; SGORBISSA, A.; VERNAZZA, T.; ZACCARIA, R. Analysis of human behavior recognition algorithms based on acceleration data. In: IEEE. **2013 IEEE International Conference on Robotics and Automation**. [*S. l.*], 2013. p. 1602–1607.

BUCCHIARONE, A.; CICCHETTI, A.; SANCTIS, M. D. Towards a domain specific language for engineering collective adaptive systems. In: IEEE. **2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W)**. [*S. l.*], 2017. p. 19–26.

BURGER, A.; CICHIWSKYJ, C.; SCHMEISSER, S.; SCHIELE, G. The elastic internet of things-a platform for self-integrating and self-adaptive iot-systems with support for embedded adaptive hardware. **Future Generation Computer Systems**, Elsevier, v. 113, p. 607–619, 2020.

CASTRO, P. A. Souza e. Tamanho de amostra e poder para três testes não-paramétricos. 2012.

CHADEGANI, A. A.; SALEHI, H.; YUNUS, M.; FARHADI, H.; FOOLADI, M.; FARHADI, M.; EBRAHIM, N. A. A comparison between two main academic literature collections: Web of science and scopus databases. **Asian Social Science**, v. 9, n. 5, p. 18–26, 2013.

CHATTERJEE, P.; ARMENTANO, R. L. Internet of things for a smart and ubiquitous ehealth system. In: IEEE. **2015 international conference on computational intelligence and communication networks (CICN)**. [*S. l.*], 2015. p. 903–907.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002.

CLOUD, A. E. C. Amazon web services. **Retrieved November**, v. 9, n. 2011, p. 2011, 2011.

COMPUTING, A. *et al.* An architectural blueprint for autonomic computing. **IBM White Paper**, Citeseer, v. 31, n. 2006, p. 1–6, 2006.

CONTADOR, J. L.; SENNE, E. L. F. Testes não paramétricos para pequenas amostras de variáveis não categorizadas: um estudo. **Gestão & Produção**, SciELO Brasil, v. 23, p. 588–599, 2016.

COSTA, C. A. da; PASLUOSTA, C. F.; ESKOFIER, B.; SILVA, D. B. da; RIGHI, R. da R. Internet of health things: Toward intelligent vital signs monitoring in hospital wards. **Artificial intelligence in medicine**, Elsevier, v. 89, p. 61–69, 2018.

CUNHA, V. S. d. *et al.* Uma abordagem orientada a serviços para captura de métricas de processo de desenvolvimento de software. Pontifícia Universidade Católica do Rio Grande do Sul, 2006.

DASTJERDI, A. V.; BUYYA, R. Fog computing: Helping the internet of things realize its potential. **Computer**, IEEE, v. 49, n. 8, p. 112–116, 2016.

DAVID, R.; DUKE, J.; JAIN, A.; REDDI, V. J.; JEFFRIES, N.; LI, J.; KREEGER, N.; NAPPIER, I.; NATRAJ, M.; WANG, T. *et al.* Tensorflow lite micro: Embedded machine learning for tinyml systems. **Proceedings of Machine Learning and Systems**, v. 3, p. 800–811, 2021.

DERHAMY, H.; ELIASSON, J.; DELSING, J.; PRILLER, P. A survey of commercial frameworks for the internet of things. In: IEEE. **2015 ieee 20th conference on emerging technologies & factory automation (etfa)**. [*S. l.*], 2015. p. 1–8.

DÍAZ, D.; YEE, N.; DAUM, C.; STROULIA, E.; LIU, L. Activity classification in independent living environment with jins meme eyewear. In: IEEE. **2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)**. [*S. l.*], 2018. p. 1–9.

DIXIT, R.; BHUSHAN, B. Scrum: An agile software development process and metrics. **Journal on Today's Ideas-Tomorrow's Technologies**, v. 7, n. 1, p. 73–87, 2019.

DWYER, G.; AGGARWAL, S.; STOUFFER, J. **Flask: building python web services**. [*S. l.*]: Packt Publishing, 2017.

EL-RASHIDY, N.; EL-SAPPAGH, S.; ISLAM, S. R.; EL-BAKRY, H. M.; ABDELRAZEK, S. Mobile health in remote patient monitoring for chronic diseases: principles, trends, and challenges. **Diagnostics**, MDPI, v. 11, n. 4, p. 607, 2021.

GHADI, Y.; MOUAZMA, B.; GOCHOO, M.; SULIMAN, A.; TAMARA, S.; JALAL, A.; PARK, J. Improving the ambient intelligence living using deep learning classifier. In: **CMC**. [*S. l.*: *s. n.*], 2022.

GIBSON, R. M.; AMIRA, A.; RAMZAN, N.; HIGUERA, P. Casaseca-de-la; PERVEZ, Z. Multiple comparator classifier framework for accelerometer-based fall detection and diagnostic. **Applied Soft Computing**, Elsevier, v. 39, p. 94–103, 2016.

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. Internet of things (iot): A vision, architectural elements, and future directions. **Future generation computer systems**, Elsevier, v. 29, n. 7, p. 1645–1660, 2013.

GÜNDOĞRAN, C.; KIETZMANN, P.; LENDERS, M.; PETERSEN, H.; SCHMIDT, T. C.; WÄHLISCH, M. Ndn, coap, and mqtt: a comparative measurement study in the iot. In: **Proceedings of the 5th ACM Conference on Information-Centric Networking**. [*S. l.*: *s. n.*], 2018. p. 159–171.

HORTON, G. I.; RADCLIFFE, D. F. Nature of rapid proof-of-concept prototyping. **Journal of Engeering Design**, Taylor & Francis, v. 6, n. 1, p. 3–16, 1995.

HOSSAIN, T.; INOUE, S. Sensor-based daily activity understanding in caregiving center. In: IEEE. **2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)**. [*S. l.*], 2019. p. 439–440.

IFTIKHAR, M. U.; RAMACHANDRAN, G. S.; BOLLANSÉE, P.; WEYNS, D.; HUGHES, D. Deltaiot: A self-adaptive internet of things exemplar. In: IEEE. **2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. [*S. l.*], 2017. p. 76–82.

INVERARDI, P.; TIVOLI, M. The future of software: Adaptation and dependability. In: **Software engineering**. [*S. l.*]: Springer, 2007. p. 1–31.

JALAL, A.; KAMAL, S.; AZURDIA-MEZA, C. A. Depth maps-based human segmentation and action recognition using full-body plus body color cues via recognizer engine. **Journal of Electrical Engineering & Technology**, Springer, v. 14, n. 1, p. 455–461, 2019.

JALAL, A.; KAMAL, S.; KIM, D.-S. Detecting complex 3d human motions with body model low-rank representation for real-time smart activity monitoring system. **KSII Transactions on Internet and Information Systems**, v. 12, n. 3, p. 1189–1204, 2018.

JALAL, A.; KIM, Y.-H.; KIM, Y.-J.; KAMAL, S.; KIM, D. Robust human activity recognition from depth video using spatiotemporal multi-fused features. **Pattern recognition**, Elsevier, v. 61, p. 295–308, 2017.

JOSHI, A.; KALE, S.; CHANDEL, S.; PAL, D. K. Likert scale: Explored and explained. **British journal of applied science & technology**, v. 7, n. 4, p. 396–403, 2015.

JUNIOR, B. R.; ANDRADE, R. M.; MAIA, M. E.; NOGUEIRA, T. P. Succeed: Support mechanism for creating and executing workflows for decoupled sas in iot. In: IEEE. **2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)**. [*S. l.*], 2018. v. 2, p. 738–743.

JUNIOR, E. C.; MAIA, P. H. M.; AFFONSO, F. J. Sases: A framework for the development of service-based self-adaptive applications. **IEEE Latin America Transactions**, IEEE, v. 14, n. 9, p. 4187–4195, 2016.

KAMPEL, M.; DOPPELBAUER, S.; PLANINC, R. Automated timed up & go test for functional decline assessment of older adults. In: **Proceedings of the 12th EAI International Conference on Pervasive Computing Technologies for Healthcare**. [*S. l.*: *s. n.*], 2018. p. 208–216.

KASHANI, M. H.; MADANIPOUR, M.; NIKRAVAN, M.; ASGHARI, P.; MAHDIPOUR, E. A systematic review of iot in healthcare: Applications, techniques, and trends. **Journal of Network and Computer Applications**, Elsevier, v. 192, p. 103164, 2021.

KHAN, A. N.; RIZWAN, A.; AHMAD, R.; KIM, D. H. An ocf-iotivity enabled smart-home optimal indoor environment control system for energy and comfort optimization. **Internet of Things**, Elsevier, v. 22, p. 100712, 2023.

KHAN, N.; MA, Z.; ULLAH, A.; POLAT, K. Dca-iomt: Knowledge-graph-embedding-enhanced deep collaborative alert recommendation against covid-19. **IEEE Transactions on Industrial Informatics**, IEEE, v. 18, n. 12, p. 8924–8935, 2022.

KIRKWOOD, R. N.; MOREIRA, B. de S.; MINGOTI, S. A.; FARIA, B. F.; SAMPAIO, R. F.; RESENDE, R. A. The slowing down phenomenon: What is the age of major gait velocity decline? **Maturitas**, Elsevier, v. 115, p. 31–36, 2018.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering–a systematic literature review. **Information and software technology**, Elsevier, v. 51, n. 1, p. 7–15, 2009.

KODYŠ, M.; OLIVER, P.; BELLMUNT, J.; MOKHTARI, M. Human urban mobility classification in aal deployments using mobile devices. In: **Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services**. [*S. l.*: *s. n.*], 2017. p. 311–319.

KRUPITZER, C.; TEMIZER, T.; PRANTL, T.; RAIBULET, C. An overview of design patterns for self-adaptive systems in the context of the internet of things. **IEEE Access**, IEEE, v. 8, p. 187384–187399, 2020.

LAUREANO, R. *et al.* Testes de hipóteses e regressão–o meu manual de consulta rápida. **Lisboa: Edições Silabo**, 2020.

LAZAROU, I.; KARAKOSTAS, A.; STAVROPOULOS, T.; TSOMPANIDIS, T.; MEDITSKOS, G.; KOMPATSIARIS, I.; TSOLAKI, M. A novel and intelligent home monitoring system for care support of elders with cognitive impairment. **Journal of Alzheimer's Disease**, v. 54, n. 4, p. 1561–1591, 2016. Cited By 32. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84992160785&doi=10.3233%2fJAD-160348&partnerID=40&md5=047b7c8d111cfcb62cec3836426f7575. Acesso em: 1 mai. 2023.

LENG, J.; LIN, Z.; WANG, P. An implementation of an internet of things system for smart hospitals. In: IEEE. **2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)**. [*S. l.*], 2020. p. 254–255.

LI, J.; LI, M.; WANG, Z.; ZHAO, Q. An improved classification method for fall detection based on bayesian framework. In: . [*S. n.*], 2015. p. 237–242. Cited By 1. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964515019&doi=10.1109%2fROBIO.2015.7418773&partnerID=40&md5=382a7a0836f18c6f2130dc35c79d8c91. Acesso em: 1 mai. 2023.

LI, S.; MAN, C.; SHEN, A.; GUAN, Z.; MAO, W.; LUO, S.; ZHANG, R.; YU, H. A fall detection network by 2d/3d spatio-temporal joint models with tensor compression on edge. **ACM Transactions on Embedded Computing Systems**, ACM New York, NY, v. 21, n. 6, p. 1–19, 2022.

LI, W.; ZHANG, Z.; LIU, Z. Action recognition based on a bag of 3d points. In: IEEE. **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops**. [*S. l.*], 2010. p. 9–14.

LINHARES, I.; ANDRADE, R.; JUNIOR, E. C.; OLIVEIRA, P. A.; OLIVEIRA, B.; AGUILAR, P. Lessons learned from the development of mobile applications for fall detection. In: **GLOBAL HEALTH 2020, The Ninth International Conference on Global Health Challenges**. [*S. l.*]: ThinkMind, 2020. p. 18–25.

LUKASZEWSKI, A.; REYNOLDS, A. **MySQL for Python**. [*S. l.*]: Packt Publishing Ltd, 2010.

MAGNO, N. M.; ROCHA, L. C. N. da; ARAÚJO, A. P. M. de; SILVA, M. C. R. da; PINTO, D. da S.; CARDOSO, B. A.; DIAS, G. A. da S. *et al.* Relação da função vesical e marcha em indivíduos com vírus linfotrópico de células t humana tipo 1. **Saúde e Pesquisa**, v. 11, n. 2, p. 213–221, 2018.

MAIA, M. E.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. M. Loccam-loosely coupled context acquisition middleware. In: **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. [*S. l.*: *s. n.*], 2013. p. 534–541.

MAJUMDER, S.; MONDAL, T.; DEEN, M. J. A simple, low-cost and efficient gait analyzer for wearable healthcare applications. **IEEE Sensors Journal**, IEEE, v. 19, n. 6, p. 2320–2329, 2019.

MAKHLOUF, A.; BOUDOUANE, I.; SAADIA, N.; CHERIF, A. R. Ambient assistance service for fall and heart problem detection. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–20, 2018.

MARIKYAN, D.; PAPAGIANNIDIS, S. Technology acceptance model: A review. **TheoryHub Book. http://open. ncl. ac. uk**, 2022.

MARQUES, M.; SIMMONDS, J.; ROSSEL, P. O.; BASTARRICA, M. C. Software product line evolution: A systematic literature review. **Information and Software Technology**, Elsevier, v. 105, p. 190–208, 2019.

MARTINEZ, L.; FERREIRA, A. **Análise de Dados com SPSS**. [*S. l.*]: Escolar editora, 2007.

MOCK, M.; SWEETING, K. *et al.* Gait and posture-assessment in general practice. **Australian family physician**, Royal Australian College of General Practitioners, v. 36, n. 6, p. 398, 2007.

MOHAMED, A. S. A.; SOTER, E. B.; SINGH, A.; RUHAIYEM, N. I. R. Gesture based help identification for hospital & elderlycare using dynamic time warping: A systematic study. In: **Proceedings of the International Conference on Video and Image Processing**. [*S. l.*: *s. n.*], 2017. p. 94–98.

MORISIO, M.; SEAMAN, C. B.; PARRA, A. T.; BASILI, V. R.; KRAFT, S. E.; CONDON, S. E. Investigating and improving a cots-based software development. In: **Proceedings of the 22nd international conference on Software engineering**. [*S. l.*: *s. n.*], 2000. p. 32–41.

MUKHERJEE, A.; GHOSH, S.; BEHERE, A.; GHOSH, S. K.; BUYYA, R. Internet of health things (ioht) for personalized health care using integrated edge-fog-cloud network. **Journal of Ambient Intelligence and Humanized Computing**, Springer, v. 12, p. 943–959, 2021.

OREIZY, P.; GORLICK, M. M.; TAYLOR, R. N.; HEIMHIGNER, D.; JOHNSON, G.; MEDVIDOVIC, N.; QUILICI, A.; ROSENBLUM, D. S.; WOLF, A. L. An architecture-based approach to self-adaptive software. **IEEE Intelligent Systems and Their Applications**, IEEE, v. 14, n. 3, p. 54–62, 1999.

ORGANIZATION, W. H. **International Classification of Functioning, Disability, and Health: Children & Youth Version: ICF-CY.** [*S. l.*]: World Health Organization, 2007.

PALUMBO, F.; ROSA, D. L.; FERRO, E. Stigmergy-based long-term monitoring of indoor users mobility in ambient assisted living environments: The doremi project approach. In: . [*S. n.*], 2017. v. 1803, p. 18–32. Cited By 7. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85016287792&partnerID=40&md5=eb7b1d7a3d4600dc0ec0ff2acb03e84d. Acesso em: 1 mai. 2023.

PARK, H.; HONG, S.; HUSSAIN, I.; KIM, D.; SEO, Y.; PARK, S. J. Gait monitoring system for stroke prediction of aging adults. In: SPRINGER. **International Conference on Applied Human Factors and Ergonomics**. [*S. l.*], 2020. p. 93–97.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V. *et al.* Scikit-learn: Machine learning in python. **the Journal of machine Learning research**, JMLR. org, v. 12, p. 2825–2830, 2011.

PEEK, S.; LUIJKX, K.; RIJNAARD, M.; NIEBOER, M.; VOORT, C. V. D.; AARTS, S.; HOOF, J. V.; VRIJHOEF, H.; WOUTERS, E. Older adults' reasons for using technology while aging in place. **Gerontology**, v. 62, n. 2, p. 226–237, 2016. Cited By 120. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84959193555&doi=10.1159%2f000430949&partnerID=40&md5=360004e5d46033a261cf11da35c59a68. Acesso em: 1 mai. 2023.

PEGORARO, R. A. Métricas de avaliação para abordagens ágeis em projetos de software. 2014.

PEIMANKAR, A.; WINTHER, T. S.; EBRAHIMI, A.; WIIL, U. K. A machine learning approach for walking classification in elderly people with gait disorders. **Sensors**, MDPI, v. 23, n. 2, p. 679, 2023.

PREE, W. Framework development and reuse support. **Visual Object-Oriented Programming, Concepts and Environments. M. Burnett, A. Goldberg, T. Lewis (eds.). Manning-Prentice Hall**, v. 11, 1995.

PREE, W. Hot-spot-driven framework development. **Framework**, v. 2, p. B1, 2000.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [*S. l.*]: McGraw Hill Brasil, 2016.

ROBBINS, T. D.; KEUNG, S. N. L. C.; ARVANITIS, T. N. E-health for active ageing; a systematic review. **Maturitas**, Elsevier, v. 114, p. 34–40, 2018.

RODRIGUES, J. J.; SEGUNDO, D. B. D. R.; JUNQUEIRA, H. A.; SABINO, M. H.; PRINCE, R. M.; AL-MUHTADI, J.; ALBUQUERQUE, V. H. C. D. Enabling technologies for the internet of health things. **Ieee Access**, IEEE, v. 6, p. 13129–13141, 2018.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. **ACM transactions on autonomous and adaptive systems (TAAS)**, ACM New York, NY, USA, v. 4, n. 2, p. 1–42, 2009.

SANSRIMAHACHAI, W.; TOAHCHOODEE, M.; PIAKAEW, R.; VIJITPHU, T.; JEENBOONMEE, S. Real-time fall risk assessment system based on acceleration data. In: IEEE. **2017 International Conference on Orange Technologies (ICOT)**. [*S. l.*], 2018. p. 33–36.

SANTOS, E. B. d. Retake: Abordagem para teste em tempo de execução de sistemas dinamicamente adaptativos. 2020.

SANTOS, E. B. dos; ANDRADE, R. M.; SANTOS, I. de S. Runtime testing of context-aware variability in adaptive systems. **Information and Software Technology**, Elsevier, v. 131, p. 106482, 2021.

SANTOS, I. de S.; CASTRO, R. M. de; SANTOS, P. d. A. dos. **TESTDAS: Testing MEthod for Dynamically Adaptive Systems**. Tese (Doutorado) – Ph. D. Dissertation. Fortaleza, Brazil. Advisor (s) Castro Andrade, Rossana . . . , 2017.

SHULL, F.; CARVER, J.; TRAVASSOS, G. H. An empirical methodology for introducing software processes. **ACM SIGSOFT Software Engineering Notes**, ACM New York, NY, USA, v. 26, n. 5, p. 288–296, 2001.

SIEGEL, S.; JR, N. J. C. **Estatística não-paramétrica para ciências do comportamento**. [*S. l.*]: Artmed Editora, 2006.

SILVA, P. Davis' technology acceptance model (tam)(1989). **Information seeking behavior and technology adoption: Theories and trends**, IGI Global, p. 205–219, 2015.

SMITH, J. **Entity Framework core in action**. [*S. l.*]: Simon and Schuster, 2021.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Harlow, England: Addison-Wesley, 2010. ISBN 978-0-13-703515-1.

STANOJEVIĆ, V.; VLAJIĆ, S.; MILIĆ, M.; OGNJANOVIĆ, M. Guidelines for framework development process. In: IEEE. **2011 7th Central and Eastern European Software Engineering Conference (CEE-SECR)**. [*S. l.*], 2011. p. 1–9.

STATISTICS, L. Spss statistics. **SPEARMAN'S CORRELATION**, 2012.

STUDENSKI, S. **Bradypedia: is gait speed ready for clinical use?** [*S. l.*]: Springer, 2009.

SUN, F.; ZANG, W.; GRAVINA, R.; FORTINO, G.; LI, Y. Gait-based identification for elderly users in wearable healthcare systems. **Information Fusion**, Elsevier, v. 53, p. 134–144, 2020.

SUNDMAEKER, H.; GUILLEMIN, P.; FRIESS, P.; WOELFFLÉ, S. **Vision and challenges for realising the internet of things**. 2020.

SYED, L.; JABEEN, S.; MANIMALA, S.; ALSAEEDI, A. Smart healthcare framework for ambient assisted living using iomt and big data analytics techniques. **Future Generation Computer Systems**, Elsevier, v. 101, p. 136–151, 2019.

TAHIR, M.; KHAN, F.; BABAR, M.; ARIF, F.; KHAN, F. Framework for better reusability in component based software engineering. **the Journal of Applied Environmental and Biological Sciences (JAEBS)**, v. 6, p. 77–81, 2016.

TARAMASCO, C.; RODENAS, T.; MARTINEZ, F.; FUENTES, P.; MUNOZ, R.; OLIVARES, R.; ALBUQUERQUE, V. H. C.; DEMONGEOT, J. A novel low-cost sensor prototype for nocturia monitoring in older people. **IEEE Access**, IEEE, v. 6, p. 52500–52509, 2018.

TIBERGHIEN, T.; MOKHTARI, M.; ALOULOU, H.; BISWAS, J. Semantic reasoning in context-aware assistive environments to support ageing with dementia. In: SPRINGER. **International semantic web conference**. [*S. l.*], 2012. p. 212–227.

TULIO, M. Engenharia de software moderna. **Editora Independente**, 2020.

VALENTE, M. T. Engenharia de software moderna. **Princípios e Práticas para Desenvolvimento de Software com Produtividade**, v. 1, p. 24, 2020.

VASCONCELOS, D.; ANDRADE, R.; SEVERINO, V.; SOUZA, J. D. Cloud, fog, or mist in iot? that is the question. **ACM Transactions on Internet Technology (TOIT)**, ACM New York, NY, USA, v. 19, n. 2, p. 1–20, 2019.

VASYLKIV, Y.; NESHATI, A.; SAKAMOTO, Y.; GOMEZ, R.; NAKAMURA, K.; IRANI, P. Smart home interactions for people with reduced hand mobility using subtle emg-signal gestures. In: **ITCH**. [*S. l.*: *s. n.*], 2019. p. 436–443.

VIEIRA, S. M.; KAYMAK, U.; SOUSA, J. M. Cohen's kappa coefficient as a performance measure for feature selection. In: IEEE. **International Conference on Fuzzy Systems**. [*S. l.*], 2010. p. 1–8.

WEYNS, D.; SCHMERL, B.; GRASSI, V.; MALEK, S.; MIRANDOLA, R.; PREHOFER, C.; WUTTKE, J.; ANDERSSON, J.; GIESE, H.; GÖSCHKA, K. M. On patterns for decentralized control in self-adaptive systems. In: **Software Engineering for Self-Adaptive Systems II**. [*S. l.*]: Springer, 2013. p. 76–107.

WHO. The global strategy and action plan on ageing and health 2016–2020: towards a world in which everyone can live a long and healthy life. **Sixty-ninth World Health Assembly, Geneva**, p. 23–28, 2016.

WHOLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. Experimentation in software engineering: an introduction. **Massachusetts: Kluwer Academic Publishers**, 2000.

WIERINGA, R.; MORALI, A. Technical action research as a validation method in information systems design science. In: SPRINGER. **International Conference on Design Science Research in Information Systems**. [*S. l.*], 2012. p. 220–238.

WILSON, D.; WILSON, S. Writing frameworks-capturing your expertise about a problem domain. **Tutorial Notes, OOPSLA'93**, 1993.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [*S. l.*]: Springer Science & Business Media, 2012.

XIE, Y.; LI, Z.; LI, H. Analysis of software development process based on software components. In: IEEE. **2020 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA)**. [*S. l.*], 2020. p. 625–628.

YANG, Y. J.; KIM, S. Y.; CHOI, G. J.; CHO, E. S.; KIM, C. J.; KIM, S. D. A uml-based object-oriented framework development methodology. In: IEEE. **Proceedings 1998 asia pacific software engineering conference (Cat. No. 98EX240)**. [*S. l.*], 1998. p. 211–218.

YEAN, S.; LEE, B. S.; YEO, C. K.; VUN, C. H.; OH, H. L. Smartphone orientation estimation algorithm combining kalman filter with gradient descent. **IEEE journal of biomedical and health informatics**, IEEE, v. 22, n. 5, p. 1421–1433, 2018.

ZEADALLY, S.; SIDDIQUI, F.; BAIG, Z.; IBRAHIM, A. Smart healthcare: Challenges and potential solutions using internet of things (iot) and big data analytics. **PSU research review**, Emerald Publishing Limited, v. 4, n. 2, p. 149–168, 2020.

ZHANG, Z. Microsoft kinect sensor and its effect. **IEEE multimedia**, IEEE, v. 19, n. 2, p. 4–10, 2012.

APÊNDICE A – MANUAL DE EXECUÇÃO DO PROCESSO MOTION

O Manual de Execução do Processo MOTION tem por objetivo guiar a execução do processo, detalhando cada uma das suas atividades, incluindo entradas e saídas esperadas de cada atividade, uma lista de artefatos que pode ser usado para auxiliar cada atividade, os papéis do time de desenvolvimento ligados a cada atividade, uma sugestão de passos a serem seguidos em cada atividade e uma discussão sobre como cada atividade pode impactar as atividades seguintes, bem como ser impactadas pelas atividades anteriormente executadas durante o desenvolvimento do sistema.

O manual de execução do processo MOTION é apresentado a seguir.

MANUAL DE EXECUÇÃO DO PROCESSO MOTION

Atividade 1: Especificação de estados de saúde monitorados, sensores e tipo de dados de movimento

Entrada Esperada: Descrição Informal ou RFP do Sistema.
Saída Esperada: Descrição do Sistema com o detalhamento dos estados finais de saúde que se necessita monitorar para alcançar o resultado esperado, estudo de viabilidade preliminar do sistema, lista de principais sensores e informações de movimentação a serem utilizadas na aplicação.
Artefatos que podem auxiliar: Taxonomia, Grafo de Classificação, Catálogo de condições de saúde relacionados ao que se quer obter como objetivo do sistema.
Principais papéis: Gerente do Projeto ou de Negócio, Arquiteto ou Engenheiro de Software, Analista de Requisitos, P.O., Analista de Dados, Equipe de Desenvolvimento, Especialista do Domínio.

Descrição da Atividade

Nessa Atividade o time começa por especificar os estados finais que o sistema precisa monitorar ou identificar (no caso de estados de saúde anômalos) com base nos dados dos sensores, a fim de atingir os objetivos do sistema solicitados pelo cliente. Guiado pelos estados de saúde definidos, são selecionados os sensores que serão utilizados, e os possíveis padrões de movimento que podem ser identificados pelos sensores. Considerando que as aplicações que serão desenvolvidas usando o processo proposto são aplicações IoT, é esperado que sensores e possíveis atuadores presentes nos dispositivos IoT sejam utilizados, o que torna essencial essa Atividade para o processo. Vale ressaltar que essa Atividade não é simples, pois muitos dos possíveis padrões de movimentação podem ser obtidos usando mais de um tipo de sensor, como o padrão de marcha, que, por exemplo, podem usar dados de acelerômetros, giroscópios, ou sensores de pressão.

Etapas

Etapa 1: Identificar o objetivo geral e os objetivos específicos do sistema solicitado pelo cliente com base na descrição do sistema que foi fornecida.
Etapa 2: Identificar que objetivos necessitam que seja feito algum tipo de monitoramento com base em dados de sensores para ser atingido.
Etapa 3: Fazer um estudo preliminar da literatura, ou consulta aos materiais de apoio, ou ainda consulta a profissionais de saúde especializados, a fim de identificar quais possíveis estados de saúde (sintomas, padrões de movimento, condições de saúde) são necessárias serem identificadas ou monitoradas para que os objetivos sejam alcançados.

Etapa 4: Relacionar estados finais identificados na etapa anterior e avaliar as necessidades e a viabilidade de monitorar ou identificar tais estados e gerar documento final com essas informações.
Etapa 5: Analisar estados de saúde esperados e identificar padrões de movimento que podem ser utilizados para monitorar ou reconhecer esses estados.
Etapa 6: Definir features que podem ser utilizadas para analisar os padrões de movimento identificados na Atividade anterior.
Etapa 7: Selecionar os sensores que podem ser utilizados para obter as features apresentadas na Atividade anterior.
Etapa 8: Listar tipos de sensores e de dispositivos que podem ser utilizados para a aplicação.
Etapa 9: Selecionar os sensores, dispositivos e padrões de movimento a que serão monitorados pela aplicação e gerar documento contendo essas informações.

Obs.: *A consulta a um especialista de domínio da área de saúde é vantajosa para todas as Atividades do processo referentes a análise e elicitação de requisitos (Atividades 1 e 2) e Validação (Atividade 9) do processo.*

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

Os documentos gerados nessa Atividade servirão como entrada das próximas etapas do processo, porém mesmo antes de seguir para próxima Atividade, é possível discutir as informações finais geradas nesses documentos com o cliente, afim de confirmar se o cliente concorda com as informações contidas nos documentos, e se aquilo que foi planejado pela equipe como possibilidade de estados finais a serem monitorados ou identificados está de acordo com o esperado, ou se os objetivos geral e específicos devem sofrer alteração.

Caso o cliente opte por alterar os objetivos, a Atividade 1 do processo deve ser novamente executada. Caso essa alteração for uma mudança para aumentar o escopo do sistema, as demais Atividades do processo, iniciadas com base no escopo anterior podem continuar sendo executadas em paralelo e apenas as adições ao escopo seguirão para as seguintes Atividades posteriormente. Porém, se a mudança tiver grande impacto no escopo originalmente pensado, é importante que as demais Atividades em paralelo sejam interrompidas para evitar retrabalho. Então deve ser feita uma avaliação do que pode ser aproveitado dos incrementos e documentos já produzidos. Os incrementos e documentos relevantes serão analisados na Atividades 3 do processo (Análise de componentes para reúso) e reaproveitados, se possível, nas Atividades seguintes.

Um dos documentos intermediários aprovados nessa Atividade, corresponde ao estudo de viabilidade, com esse documento aprovado, devemos então seguir para as Atividades de seleção de sensores. Uma vez em posse da lista de sensores e dispositivos que podem ser utilizados, da descrição informal e da descrição dos estados de saúde é possível iniciar o esboço mais detalhado do sistema a ser desenvolvido e partir para elicitação dos demais requisitos. Nessa Atividade é possível também gerar o estudo de viabilidade, antes da elicitação detalhada dos requisitos.

Caso seja solicitado a mudança dos dispositivos e sensores após a execução dessa Atividade, contanto que os sensores e dispositivos pertençam aos mesmos tipos da lista definida na etapa 7 dessa Atividade, não haverá grande impacto para essa Atividade, porém os requisitos e o projeto da aplicação devem ser atualizados. É importante lembrar que dispositivos e sensores diferentes possuem limitações diferentes de hardware que podem impactar na qualidade dos dados, desse modo, qualquer mudança de hardware deve ser cuidadosamente analisada e o impacto maior desta mudança deve ser sentido na Atividade de projeto da aplicação, e por conseguinte, na Atividade de desenvolvimento..

Atividade 2: Especificação dos demais requisitos

Entrada Esperada: Descrição Informal (ou RFP), Documento de descrição de estados de saúde e Documento com sensores, dispositivos e padrões de movimentos selecionados.
Saída Esperada: Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de Usuário.
Artefatos que podem auxiliar: Taxonomia, Grafo de Classificação, Catálogo de condições de saúde relacionados ao que se quer obter como objetivo do sistema.
Principais papéis: Arquiteto ou Engenheiro de Software, Analista de Requisitos, P.O., Analista de Dados, Equipe de Desenvolvimento, Analista de Teste, Especialista do Domínio, Analista de UI/UX.

Descrição da Atividade

Nessa segunda Atividade do processo, é feita a elicitação dos demais requisitos específicos do tipo da aplicação solicitada pelo cliente, levando em conta também todas as regras de negócio.

Etapas

Nessa Atividade pode ser utilizado qualquer subprocesso de levantamento de requisitos conhecidos. As Atividades abaixo contemplam as Atividades macro da elicitação de requisitos à ser feita, mas a ordem em que são executadas pode divergir. Caso a equipe já siga um subprocesso próprio para elicitação de requisitos é recomendados que esse subprocesso seja utilizado para essa Atividade.

Etapa 1: Leitura dos documentos gerados até então e identificação dos principais elementos e regras de negócio do sistema.
Etapa 2: Identificação dos atores do sistema.
Etapa 3: Descrição dos requisitos funcionais divididos por atores.
Etapa 4: Elicitação dos requisitos não-funcionais e suas métricas.
Etapa 5: Homologação dos requisitos pelo P.O. e cliente.
Etapa 6: Geração de casos de uso ou estórias de usuários.
Etapa 6.1: Catalogação e geração de dicionários de termos usados nos elementos do sistema (Opcional).
Etapa 7: Finalização do documento de requisitos e documentação dos casos e usos e estórias de usuários.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

Tanto o documento de requisitos, quanto os casos de uso ou estórias de usuários geradas nessa Atividade serão a base para o projeto, desenvolvimento e validação do sistema que serão executados nas etapas seguintes do processo. Porém, não é necessário que todos os requisitos, casos de usos ou estórias de usuários tenham sido documentados para que as próximas Atividades sejam executadas, pois o processo proposto permite que as suas Atividades, a partir da quarta Atividade, correspondente a revisão dos requisitos, sejam executadas de maneira incremental, então, a cada período de tempo um pequeno número de requisitos, casos de usos ou estórias de usuários são projetados, desenvolvidos e validados gerando ao fim um novo incremento e uma nova versão estável do sistema.

Logo, se for necessário alguma mudança em um requisito, essa mudança será contemplada em uma ou mais Atividades em um ciclo futuro de desenvolvimento gerando o mínimo possível impacto no processo e no sistema como um todo. Há exceções onde a mudança em um requisito, ou conjunto de requisitos, é muito grande, gerando assim um grande impacto. No entanto, esse tipo de risco deve ser gerenciado continuamente e pode ser mitigado com o contato recorrente com o cliente ao longo do projeto, de modo a minimizar a chance de que uma mudança muito grande seja requisitada de maneira inesperada.

Atividade 3: Análise de Componentes para Reúso Considerando o Loop de Adaptação

Entrada Esperada: Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de Usuário.
Saída Esperada: Lista de componentes de reúso a serem utilizados.
Artefatos que podem auxiliar: Lista de componentes de reúso para *softwares* autoadaptativos conhecidos .
Principais papéis: Arquiteto ou Engenheiro de Software, P.O., Analista de Dados, Equipe de Desenvolvimento, Analista de Teste, Especialista do Domínio.

Descrição da Atividade

Nessa terceira Atividade do processo, é feita a análise de componentes de reúso (modelos, plataformas, bibliotecas e frameworks), que podem ser utilizados para auxiliar o desenvolvimento da aplicação de IoHT autoadaptativa.

Etapas

Nessa Atividade pode ser utilizado qualquer subprocesso de levantamento de componentes de reúso. Durante esta Atividade, é importante identificar componentes que possam auxiliar no desenvolvimento do aplicativo e que possam ser adicionados ao ciclo de adaptação. Nesse momento é fundamental a presença do arquiteto ou do responsável pela arquitetura da aplicação, pois a decisão de usar um componente ou outro de reúso pode impactar diretamente na arquitetura.

Etapa 1: Identificação de componentes de reúso que podem ser úteis para auxiliar o desenvolvimento da aplicação de acordo com os requisitos solicitados. Nessa etapa pode ser feita uma busca pela literatura ou por componentes de reúso conhecidos utilizados para o desenvolvimento de aplicações com requisitos similares aos elicitados anteriormente. O especialista de domínio pode ajudar também com recomendações de componentes de reúso que já utilizou ou são de seu conhecimento.
Etapa 2: Verificação dos componentes com relação ao *loop* de adaptação. Nessa etapa é verificados se os componentes identificados não inviabilizam o *loop* de adaptação e em que fases do *loop* podem ser agregados.
Etapa 3: Seleção dos componentes de reúso que serão utilizados.
Etapa 4: Geração de lista de componentes de reúso selecionados.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

Ao final dessa Atividade é gerada a lista de componentes de reúso que serão utilizados. Nesse momento em paralelo, espera-se que a arquitetura do sistema já comece a ser projetada considerando os componentes, embora ainda seja importante executar parcial ou completamente as Atividades 4 (Adaptação de requisitos e criação da lista de possíveis adaptaçãoes) e 5 (Criação das regras de adaptação) antes de finalizar a arquitetura do *software*, uma vez que as decisões tomadas nessas etapas podem impactar na construção desse artefato. Caso seja identificado apenas durante a Atividade 6 (Projeto com reúso considerando os componentes do MAPE-K) que os componentes escolhidos durante a Atividade 3 não podem ser usados ou que não agregam o suficiente à aplicação, é importante que as Atividades 3 a 5 sejam revistas, pois a decisão por um ou outro componente de reúso pode impactar em alterações nos requisitos e regras de adaptação, uma vez que o componente pode está diretamente ligada a algum requisito elicitado ou regra de adaptação planejada.

Atividade 4: Adaptação de Requisitos e criação das lista de possíveis adaptações

Entrada Esperada: Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de Usuário, Lista de componentes de reúso a serem utilizados, Projeto do Software atualizada na iteração anterior (se houver) e Relatório de testes atualizado na iteração anterior (se houver).
Saída Esperada: Documento de Elicitação de Requisitos atualizado, Documentação de Casos de uso ou Estórias de usuário atualizados, Lista de possíveis adaptações a serem implementadas, Lista de Casos de uso ou Estórias de usuários a serem implementados nessa iteração priorizada, Plano de Testes atualizado e Plano de Avaliação de Qualidade do Software
Artefatos que podem auxiliar: Taxonomia, Grafo de Classificação, Template de Regras de Adaptação.
Principais papéis: Arquiteto ou Engenheiro de Software, Analista de Requisitos, P.O., Analista de Dados, Equipe de Desenvolvimento, Equipe de Testes, Especialista do Domínio, Analista de UI/UX, Analista de Qualidade (opcional, mas altamente recomendável).

Descrição da Atividade

Nessa quarta Atividade do processo, é feita a atualização dos requisitos para melhor adequar os componentes de reúso escolhidos e para agregar qualquer tipo mudança a ser feita nos requisitos. Com a atualização dos requisitos, também pode ser necessário atualizar os casos de uso ou estórias de usuários. Além disso, nessa Atividade do Processo deve ser gerada ou atualizada a lista com as possíveis adaptações da aplicação de IoHT autoadaptativa. Também durante essa Atividade, a equipe de testes deve trabalhar na cosntrução ou atualização do Plano de testes. Por fim, caso seja de interesse da equipe utilizar um método iterativo, por exemplo o SCRUM, para a implementação, essa etapa pode ser vista como a primeira da iteração e nela deve ser feita a seleção dos casos de uso ou estórias de usuário que serão implementados na iteração atual.

Etapas

A partir dessa Atividade pode ser utilizado um método iterativo, como o Scrum, em coonjunto com o processo de desenvolvimento MOTION. As Atividades abaixo contemplam as Atividades macro da revisão dos requisitos e seleção de casos de uso ou estórias de usuários a serem implmentadas em cada iteração, mas a ordem em que são executadas pode divergir. Caso a equipe já siga um subprocesso próprio para revisão de requisitos é recomendado que esse subprocesso seja utilizado. Além da revisão dos requitisos também é feita (na etapa 6) a geração ou revisão da lista de possíveis adaptações da aplicação IoHT autoadaptativa.

Etapa 1: Revisão dos documentos gerados até então e identificação dos principais elementos e regras de negócio do sistema.
Etapa 2: Revisão dos atores do sistema.
Etapa 3: Adição ou revisão de requisitos para contemplar o uso dos componentes de reíso selecionados na Atividade anterior.
Etapa 4: Revisão dos requisitos funcionais divididos por atores.
Etapa 5: Revisão dos requisitos não-funcionais e suas métricas.
Etapa 6: Geração´ou revisão da lista de possíveis adaptações da aplicação.
Etapa 7: Geração de casos de uso ou estórias de usuários .
Etapa 7.1: Revisão do dicionários de termos usados nos elementos do sistema (Opcional).
Etapa 8: Atualização do documento de requisitos e documentação dos casos e usos e estórias de usuários.
Etapa 9.1: Homologação e priorização dos casos e usos ou estórias de usuários pelo P.O.
Etapa 9.2: Criação ou atulização do plano de testes pela equipe de testes (Essa etapa ocorre em paralelo com as etapas 9.1, 9.3 e 10).
Etapa 9.3: Criação ou atualização do plano de avaliação de qualidade do sistema (Essa etapa ocorre em paralelo com as etapas 9.1, 9.2 e 10) (opcional, mas altamente recomendável)
Etapa 10: Seleção dos casos de uso ou estórias de usuário que serão implementados nessa iteração.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

O documento de requisitos e os casos de uso ou estórias de usuários atualizados nessa Atividade, bem como a lista de possíveis adaptações, serão a base para a criação das regras de adaptação, projeto, desenvolvimento e validação do sistema que serão executados nas Atividades seguintes do processo. Porém, vale relembrar que, não é necessário que todos os requisitos, casos de usos ou estórias de usuários tenham sido documentados para que as próximas Atividades sejam executadas, pois o processo proposto permite que as suas Atividades, a partir desta quarta Atividade, sejam executadas de maneira incremental, desse modo, a cada período de tempo um pequeno número de requisitos, casos de uso ou estórias de usuários são projetados, desenvolvidos e validados gerando ao fim um novo incremento e uma nova versão estável do sistema autoadaptativo.

Assim, o processo incorpora as vantagens de desenvolvimento de sistemas utilizando os modelos de processos incremental e evolucionário em tempo de projeto (quando o sistema ainda não foi disponibilizado para execução pelo usuário). Dessa forma, a cada ciclo (ou iteração) de desenvolvimento um conjunto de requisitos e casos de usos ou estórias de usuários são implementadas naquela iteração. Ao fim desse ciclo, uma nova versão do sistema, contemplando os casos de usos ou estórias de usuários selecionados é gerada e testada. Logo, se for necessário alguma mudança em um requisito, essa mudança será contemplada em uma ou mais Atividades em um ciclo futuro de desenvolvimento gerando o menor impacto possível no processo e no sistema autoadaptativo como um todo. Pode ocorrer que uma mudança em um requisito, ou

conjunto de requisitos, seja muito grande, gerando assim um grande impacto. No entanto, esse tipo de risco precisa ser gerenciado continuamente, o que pode ser feito, por exemplo, ao manter contato recorrente com o cliente ao longo do projeto, de modo a minimizar a chance de que uma mudança muito grande seja requisitada de maneira inesperada.

Atividade 5: Criação das Regras de Adaptação

Entrada Esperada: Documento de Elicitação de Requisitos, Lista de possíveis adaptações.
Saída Esperada: O conjunto de regras de adaptação e Arquivo com regras de adaptação estruturadas (opcional).
Artefatos que podem auxiliar: Template de Regras de Adaptação.
Principais papéis: Arquiteto ou Engenheiro de Software, Analista de Requisitos, Analista de Dados, Equipe de Desenvolvimento, Especialista do Domínio.

Descrição da Atividade

Nessa quinta Atividade do processo é feita a criação e atualização das regras de adaptação.

Etapas

Nessa Atividade ao criar as regras de adaptação é interessante que seja criado um formato específico também para a documentação de cada regra de adaptação, a fim de, se necessário, ser possível gerar novas regras de adaptação durante a penúltima Atividade do processo, referente a Evolução durante a execução do *software*. Para auxiliar essa Atividade, você pode utilizar um template já existente para declaração de regras de adaptação que indique as ações que devem ser executadas, os contextos relacionados a essas ações e as regras relacionadas a esses contextos, que identifiquem que a ação deve ou não ser executada.

Etapa 1: Revisão das listas de possíveis adaptações.
Etapa 2: Identificação dos contextos possíveis de ser monitorados.
Etapa 3: Seleção das adaptações que podem ser executadas considerando os contextos que podem ser monitorados.
Etapa 4: Geração das regras de mudança de contexto que irão gerar as adaptações.
Etapa 5: Documentação do conjunto de regras de adaptação.
Etapa 6: Geração do arquivo estruturado com as regras de adaptação (opcional).

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

O conjunto de regras de adaptação são necessários para que durante o projeto de sistema seja projetado também o mecanismo de adaptação que irá seguir as regras geradas. Caso seja necessário fazer mudanças ou adicionar adaptações em tempo de projeto é possível atualizar o conjunto de regras de adaptação em um nova iteração. Espera-se que também seja possível atualizar as adaptações durante o sistema já em execução, desse modo, é importante que a

estrutura das regras seja clara e que haja um mecanismos que permita adição de novas regras de adaptação durante a Atividade de Evolução.

Atividade 6: Projeto com Reúso

Entrada Esperada: Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de Usuário, Lista de Casos de uso ou Estórias de usuários a serem implementados nessa iteração, Lista de componentes de reúso a serem utilizados, Conjunto de regras de adaptação, Arquivo com regras de adaptação estruturadas (opcional), Versão atual do *software* desenvolvido nas iterações anteriores (se houver), Arquitetura do Software atualizada na iteração anterior (se houver) e Projeto do Software atualizado na iteração anterior (se houver).
Saída Esperada: Arquitetura do Software atualizada, Projeto do Software atualizado, Diagramas UML (opcional).
Artefatos que podem auxiliar: Grafo de Classificação, Modelo ou meta-modelo de desenvolvimento da aplicação autoadaptativa, Frameworks para aplicações de IoHT autoadaptativas e Middlewares de IoT.
Principais papéis: Arquiteto ou Engenheiro de Software, Analista de Dados, Equipe de Desenvolvimento, Equipe de Testes, Especialista do Domínio, Analista de UI/UX.

Descrição da Atividade

Nessa sexta Atividade do processo é projetada ou atualizada a arquitetura e é feito o projeto do que será implementado durante a iteração atual do desenvolvimento do sistema. Durante essa Atividade é importante ter em mente que o sistema deve ser projetado de modo a permitir a adaptação em tempo de execução considerando as regras de adaptação. Uma opção interessante é o projeto de ações que podem ou não ser executadas dado a análise e o planejamento executados durante o ciclo de adaptação. Outra possibilidade é o uso de serviços como ações. Enfim, a maneira como o sistema irá lhe dar com a adaptação deve ser projetado com bastante cuidado.

Etapas

Nessa Atividade ao gerar a arquitetura do *software* e projetar os elementos que serão implementados na iteração atual é fundamental que o que for projetado leve em consideração o ciclo de adapção MAPE-K. É importante que o ciclo de adaptação execute como um subprocesso (*thread*) em paralelo ao funcionamento padrão do sistema, uma vez que o mesmo deve está relacionado as possíveis adaptações e, desse modo, não deve intereferir a todo momento no funcionamento do sistema. É recomendado que cada etapa do ciclo MAPE-K seja implementado como um módulo separado.

Etapa 1: Criar ou atualizar a artuiterura do sistema
Etapa 2: Criar ou atualizar o projeto do mecanismo de aramazenamento de dados.

Etapa 3: Criar ou atualizar o projeto do mecanismo (*middleware* ou mecanimso próprio) de coletada de dados de sensores e manipulação de atuadores.
Etapa 4: Criar ou atualizar o projeto do módulo de monitoramento do contexto.
Etapa 5: Criar ou atualizar o projeto do módulo de análise do contexto monitorado.
Etapa 6: Criar ou atualizar o projeto do módulo de planejamento (lembrando que adaptações devem ser planejadas, caso necessário).
Etapa 7: Criar ou atualizar o projeto do módulo de execução.
Etapa 8: Criar ou atualizar o projeto do gerenciador da base de conhecimento do ciclo de adaptação.
Etapa 9: Projetar as Atividades refentes aos casos de uso ou estórias de usuários a serem implementadas na iteração atual.
Etapa 10: Projetar as "telas" do sistema (time de UX/UI).

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

A arquitetura e o projeto dos diversos módulos do sistema, em especial dos casos de uso ou estórias de usuário que irão ser desenvolvidos na iteração atual, são a base para o que deve ser desenvolvido na Atividade seguinte do processo. Vale sempre lembrar que o projeto deve sempre levar em consideração, que uma vez em execução o sistema deve ser capaz de se autoadaptar as mudanças no contexto. Além disso, é importante que o projeto considere como o sistema poderá evoluir durante sua execução com base principamente na adição ou seleção de possíveis ações e adaptações, o que só será possível se o sistema for bem projetado para que essas novas ações e adaptações possam ser incorporadas sem interferir no funcionamento do que já foi projetado e que já estará em execução.

Atividade 7: Implementação e Integração

Entrada Esperada: Versão atual do *software* implementado nas iterações anteriores (se houver), Documentação de Casos de uso ou Estórias de usuário, Lista de Casos de uso ou Estórias de usuário a serem implementados nessa iteração, Conjunto de regras de adaptação, Arquivo com regras de adaptação estruturadas (opcional), Arquitetura do Software, Projeto do Software e Diagramas UML (opcional).
Saída Esperada: Versão do *software* com o que foi implementado nas iterações anteriores (se houver) integrada com os novos casos de uso ou estórias de usuários implementadas ou parcialmente implementadas nessa iteração.
Artefatos que podem auxiliar: Grafo de Classificação, Modelo ou meta-modelo de desenvolvimento da Aplicação Autoadaptativa, Frameworks para aplicações de IoHT autoadaptativas e Middlewares de IoT.
Principais papéis: Analista de Dados, Equipe de Desenvolvimento, Equipe de Testes, Especialista do Domínio.

Descrição da Atividade

Nessa sétima Atividade do processo serão desenvolvidos os módulos e os casos de uso ou estórias de usuário, bem como as telas projetadas para esses, projetados na etapa anterior dessa ou de iterações anteriores, considerando os casos de uso ou estórias de usuário selecionados/priorizados para execução nessa iteração.

Etapas

Nessa Atividade são implementados os componentes do *software* selecionados/priorizados para execução nessa iteração, incluindo tanto o *back-end*, quanto *front-end*. Também é importante que os componentes implementados sejam integrados ao que já foi implementado nas iterações anteriores do processo (se existir).

Etapa 1: Atribuição aos membros do time de desenvolvimento as Atividades que irão implementar nessa iteração.
Etapa 2: Implementação das Atividades pelos membros do time de desenvolvimento (em paralelo para cada subgrupo do time de desenvolvimento que estiver trabalhando nas Atividades).
Etapa 3: Desenvolvimento dos testes unitários da Atividade implementadas pelos membros do time de desenvolvimento (em paralelo para cada subgrupo do time de desenvolvimento que estiver trabalhando nas Atividades).

Etapa 4: Escolha de uma nova Atividade, caso ainda haja Atividades a serem implmentadas nessa iteração do processo e em seguida, volta-se a etapa 2 (em paralelo para cada subgrupo do time de desenvolvimento que estiver trabalhando nas Atividades).
Etapa 5: Integração das Atividades implementadas nessa iteração a versão estável do sistema implementadas nas iterações anteriores (se houver).
Etapa 6: Geração da nova versão entregável (estável) do sistema.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

A versão atual do sistema implementada nessa iteração será usada como base para validação e verificação, que será feita na próxima Atividade, e como base para as Atividades de revisão de requisitos e projeto do *software*, que serão executadas na próxima iteração, caso a iteração atual não seja a última.

Atividade 8: Validação e Verificação (Em tempo de projeto)

Entrada Esperada: Versão atual do *software*, Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de Usuário, Lista de Casos de uso ou Estórias de usuários a serem implementados nessa iteração, Arquitetura do Software, Projeto do Software, Relatório de testes atualizado na iteração anterior (se houver), Relatório de Avaliação de Qualidade atualizado na iteração anterior (se houver), Plano de Avaliação de Qualidade do Software (se houver), Conjunto de regras de adaptação, Arquivo com regras de adaptação estruturadas (opcional) e Processo Automatizado de Avaliação do Sistema durante a execução atualizado (Se já existir).
Saída Esperada: Plano de testes atualizado, Relatório de testes atualizado, Relatório de Avaliação de Qualidade atualizado e Processo Automatizado de Avaliação do Sistema durante a Execução atualizado.
Artefatos que podem auxiliar: Modelo ou meta-modelo de desenvolvimento do sistema autoadaptativo.
Principais papéis: P.O, Equipe de Desenvolvimento, Equipe de Testes, Analista de UI/UX, Analista de Qualidade.

Descrição da Atividade

Nessa oitava Atividade do processo é feita a verificação e validação da versão atual do *software* implementada nessa iteração. Além disso, é feita a avaliação da qualidade da versão atual do *software*. Também deve ser planejado e desenvolvido um processo automatizado de validação do sistema em execução.

Etapas

Nessa Atividade são atualizados os planos de teste, incluindo os casos e procedimentos de teste. São também executados os testes projetados para os casos de uso ou estórias de usuários já implementadas e integradas a versão atual do *software*. É então elaborado o relatório de testes, que indicará quais funcionlidades estão funcionando corretamente e quais devem ser reexaminadas e sofrer atualizações em iterações posteriores. Também é nesse momento que deve ser feita a avaliação da qualidade e das caracteristicas de qualidade essenciais para o *software* que está sendo desenvolvido. Além disso, deve ser planejado e desenvolvido um processo automatizado de validadação do sistema em execução.

Etapa 1: Atualização do Plano de Testes.
Etapa 2: Execução dos testes.
Etapa 3: Homologação da versão atual do software pelo cliente (opcional, mas altamente recomendável).

Etapa 4: Geração ou atualização do Relatório de Testes.
Etapa 5: Avaliação de qualidade da versão atual do *software* (opcional, mas altamente recomendável).
Etapa 6: Geração ou atualização do Relatório de Qualidade.
Etapa 7: Projeto e Desenvolvimento do processo automatizado de validação da versão atual do sistema em execução. Essa etapa pode ser executada em paralelo com as demais etapas dessa Atividade.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

Tanto o plano de teste, quando o relatório de testes e o relatório de avaliação de qualidade serão usados também como base para escolha das Atividades e casos de uso ou estórias de usuário que devem ser selecionados para projeto e implementação na próxima iteração. E para geração dos relatórios da versão final do software.

As próximas duas Atividades do processo MOTION serão executadas apenas quando o sistema estiver em execução e continuarão sendo executadas continuamente enquanto persistir a vida útil do sistema.

Atividade 9: Evolução

Entrada Esperada: Software Desenvolvido, Documento de Elicitação de Requisitos, Documentação de Casos de uso ou Estórias de usuário, Conjunto de regras de adaptação, Arquivo com regras de adaptação estruturadas (opcional).
Saída Esperada: Nova versão do Software desenvolvido, Novas possíveis adaptações para o software desenvolvido.
Artefatos que podem auxiliar: Modelo ou meta-modelo de desenvolvimento da aplicação autoadaptativa e Template de Regras de Adaptação.
Principais papéis: Analista de Dados, Equipe de Desenvolvimento, Equipe de Testes, Especialista de Domínio, Analista UI/UX.

Descrição da Atividade

Nessa nona Atividade do processo são projetados novas ações e adaptações que podem ser adicionadas ao *software* com o sistema já em execução sem interferir, ou interferindo o mínimo possível no sistema em execução. É importante que durante as Atividades 6 e 7 de projeto do *software* e desenvolvimento do *software* sejam elaborados mecanismos que permitam que essas novas adaptações e ações sejam adicionadas durante a execução do sistema, isso pode ser feito por exemplo, gerando um padrão ou template para as ações e adaptações e uma maneira (por exemplo, usando um arquivos, e/ou serviços) que identifique as ações e adaptações que atualmente podem ser executadas pelo sistema. Vale destacar que o sistema já deve ser projetado com um conjunto de adaptações possíveis durante a execução. Essa ações projetadas em tempo de projeto, podem ser atualizadas durante essa Atividade, mas não são as mesmas das que podem ser projetadas durante essa Atividade de Evolução.

Etapas

Nessa Atividade são projetados novas ações e adaptações ou atualizadas as adaptções já existentes, que serão integradas a versão atual do software com o mesmo em execução sem interferir na execução atual do mesmo

Etapa 1: Projeto das novas adaptações ou da atualização das adaptações existentes.
Etapa 2: Projeto das novas ações.
Etapa 3: Desenvolvimento das novas ações.
Etapa 4: Desenvolvimento de testes unitários para as novas ações.

Etapa 5: Integração das novas ações e adaptações, ou das adaptações atualizadas, ao sistema em execução, usando o mecanismo gerado em tempo de projeto para prover essa integração.

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

A nova versão do sistema gerada durante essa Atividade será usado como base para a próxima Atividade de Validação em tempo de execução. *É importante destacar que a próxima Atividade do processo não necessariamente é iniciada apenas depois de uma Evolução, mas também pode ser inciada depois que uma adaptação projetada em tempo de projeto seja executada.*

Atividade 10: Validação e Verificação (Sistema em execução)

Entrada Esperada: Versão atual do *software*, Adaptações do *software* desenvolvido, Conjunto de regras de adaptação, Arquivo com regras de adaptação estruturadas (opcional) e Processo Automatizado de Avaliação do Sistema durante a Execução.
Saída Esperada: Relatório de testes com sistema em execução.
Artefatos que podem auxiliar: Abordagem para testes de sistemas autoadaptativos em execução
Principais papéis: Analista de Dados, Equipe de Desenvolvimento, Equipe de Testes, Especialista do Domínio.

Descrição da Atividade

Nessa décima Atividade do processo é feita a validação e verificação do sistema durante a execução para garantir o correto funcionamento do sistema após as adaptações serem executadas e garantir que o sistema continua funcionando corretamente após as novas adaptações e ações projetadas na Atividade de Evolução serem adicionadas ao *software*.

Etapas

Nessa Atividade é validado o sistema durante a execução caso ocorra uma das duas seguintes coisas: (1) após o sistema em execução se adaptar, ou (2) após o sistema Evoluir (novas adaptações e ações serem adicionadas ao sistema durante a execução do mesmo).

Etapa 1: Execução do processo automatizado de avaliação do sistema em tempo de execução
Etapa 2: Produção do relatório de testes com sistema em execução

Como a saída dessa Atividade impacta nas próximas e qual o impacto de mudanças nessa Atividade

O relatório gerado por esse processo pode ser avaliado e **se for indicado** mal funcionamento do sistema por algum motivo, deve ser então avaliado o desenvolvimento de uma nova versão do sistema em paralelo a execução do sistema atual. Também se necessário, o sistema atual deve ser modificado para ser colocada em execução imediatamente uma versão estável anterior do sistema. As ações e adaptação adicionadas posteriormente a essa versão estável devem ser adicionadas na nova versão do sistema. Quando a nova versão estável do sistema estiver pronta, ela deve substituir a versão estável atual do sistema. Caso não haja mal funcionamento do sistema ou o mal funcionamento possa ser resolvido com adição de novas adaptações e ações durante a Atividade Evolução, o sistema continuará executando.

APÊNDICE B – TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO (TCLE)

Caro participante,

Você está sendo convidado pelo pesquisador Evilasio Costa Junior, doutorando pelo Programa de Mestrado e Doutorado (MDCC) da Universidade Federal do Ceará (UFC) e orientado pela professora Dra. Rossana Maria de Castro Andrade e co-orientado pelo professor Dr. Leonardo Sampaio Costa, para participar da pesquisa intitulada "MOTION - Processo de Desenvolvimento de Aplicações de Internet of Health Things Auto Adaptativas Baseadas em Padrões de Movimento".

Você não deve participar contra a sua vontade. Por favor, leia atentamente as informações a seguir e faça qualquer pergunta que desejar, para que todos os procedimentos desta pesquisa sejam esclarecidos.

O referido estudo visa propor um processo de desenvolvimento de aplicações autoadaptativas de saúde que utilizam dispositivos inteligentes, as quais se inserem dentro do contexto da Internet das coisas médicas, no inglês Internet of Health Things - IoHT, baseadas em padrões de movimento, intitulado MOTION, e um conjunto de artefatos que auxiliem a execução desse processo. Essa investigação é relevante porque há uma relação direta entre padrões de movimento e condição de saúde do paciente. Além disso, o monitoramento desses padrões de movimento pode auxiliar na detecção antecipada de problemas de saúde. No entanto, a maioria das aplicações de saúde desenvolvidas com base em padrões de movimento não utilizam um processo de desenvolvimento preparado para as especificidades dessas aplicações, muitas não possuem a característica de auto adaptação. A falta de um processo de desenvolvimento adequado dificulta o gerenciamento do desenvolvimento e dos riscos envolvidos, e pode acabar por impactar na qualidade da aplicação. Dessa forma, pretendemos avaliar o uso do processo proposto o para o desenvolvimento do tipo de aplicação supracitado por profissionais de Tecnologia de Informação e Comunicação (TIC) e os benefícios do uso desse processo e dos artefatos de software propostos segundo esses profissionais.

Para alcançar o objetivo descrito acima, pretende-se acompanhar o desenvolvimento de aplicações IoHT autoadaptativas baseadas em dados do movimento por grupos de profissionais de TIC e coletar feedbacks e dados durante o desenvolvimento que nos permitam avaliar o a utilização do processo proposto por esses profissionais em comparação a uso de processos já consolidados de desenvolvimentos de softwares gerais. Durante todo o período de coleta de dados da avaliação (6 semanas), as equipes de profissionais de TIC serão acompanhados e monitorados.

Antes e após esse período de coleta de dados da avaliação, os profissionais de TIC também serão submetidos a questionários para identificação do perfil dos profissionais e coleta de seus feedback sobre o uso do processo proposto. Cada profissional de TIC convidados deve concordar em responder os dois questionários supracitados e participar durante o período de coleta de dados da avaliação (6 semanas) do desenvolvimento das aplicações que serão acompanhadas e monitoradas.

NÃO haverá riscos de danos à saúde física dos participantes e os possíveis desconfortos em relação ao fornecimento de dados serão minimizados pela completa anonimização dos dados. Por fim, destacamos que os participantes NÃO receberão nenhum pagamento por participar da pesquisa.

Sua participação é fundamental para possibilitar esse avanço na ciência. Vale ressaltar que, a qualquer momento, você poderá se recusar a continuar participando da pesquisa e também poderá retirar o consentimento, sem que isso lhe traga qualquer prejuízo. Além disso, as informações conseguidas por meio de sua participação NÃO permitirão a identificação da sua pessoa, exceto aos responsáveis pela pesquisa. Por fim, informamos que a divulgação dos dados só será feita entre os estudiosos do assunto.

Endereço do responsável pela pesquisa:

Nome: Evilasio Costa Junior Instituição: Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat) da Universidade Federal do Ceará (UFC) Endereço: Campus do Pici, Bloco 942-A, UFC, Fortaleza, Brasil Telefone/e-mail para contato: (85) 9.8934-4291 / evilasiojunior@great.ufc.br

O abaixo assinado , _____ , ____ anos, RG _____ , declara que é de livre e espontânea vontade que está como participante de uma pesquisa. Eu declaro que li cuidadosamente este Termo de Consentimento Livre e Esclarecido e que, após sua leitura, tive a oportunidade de fazer perguntas sobre o seu conteúdo, como também sobre a pesquisa, e recebi explicações que responderam por completo minhas dúvidas. E declaro, ainda, estar recebendo uma via assinada deste termo.

Fortaleza (CE), ____/____/____

_____ ____/____/____ _____

Participante da pesquisa Data Assinatura

_____	____/____/____	_____
Nome do pesquisador	Data	Assinatura
_____	____/____/____	_____
profissional que aplicou o TCLE	Data	Assinatura