Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Kathiani Elisa de Souza

Uma Abordagem para Validação e Recuperação de Erros em Dados de Sensores em Aplicações para Cidades Inteligentes

São Carlos
2024

Kathiani Elisa de Souza

Uma Abordagem para Validação e Recuperação de Erros em Dados de Sensores em Aplicações para Cidades Inteligentes

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutora em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Fabiano Cutigi Ferrari

São Carlos
2024

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Relatório de Defesa de Tese
Candidata: Kathiani Elisa de Souza

Aos 10/12/2024, às 08:00, realizou-se na Universidade Federal de São Carlos, nas formas e termos do Regimento Interno do Programa de Pós-Graduação em Ciência da Computação, a defesa de tese de doutorado sob o título: Uma abordagem para validação de dados de sensores em aplicações para cidades inteligentes, apresentada pela candidata Kathiani Elisa de Souza. Ao final dos trabalhos, a banca examinadora reuniu-se em sessão reservada para o julgamento, tendo os membros chegado ao seguinte resultado:

Participantes da Banca	**Função**	**Instituição**	**Resultado**	**Resultado Final**
Prof. Dr. Fabiano Cutigi Ferrari	Presidente	UFSCar	Aprovado	**Aprovado**
Prof. Dr. Daniel Lucrédio	Titular	UFSCar	Aprovado	
Prof. Dr. Nelio Alessandro Azevedo Cacho	Titular	UFRN	Aprovado	
Prof. Dr. Delano Medeiros Beder	Titular	UFSCar	Aprovado	
Prof. Dr. Jó Ueyama	Titular	ICMC/USP	Aprovado	

Parecer da Comissão Julgadora*:

A aluna apresentou o trabalho de forma clara e consistente com o texto entregue à Comissão Avaliadora. Ouviu as considerações feitas pelos membros da comissão e respondeu aos questionamentos a ela feitos. Se comprometeu a incorporar no texto as correções e sugestões apresentadas pelos membros da comissão.

Encerrada a sessão reservada, o presidente informou ao público presente o resultado. Nada mais havendo a tratar, a sessão foi encerrada e, para constar, eu, Ivan Rogério da Silva, representante do Programa de Pós-Graduação em Ciência da Computação, lavrei o presente relatório, assinado por mim e pelos membros da banca examinadora.

Prof. Dr. Fabiano Cutigi Ferrari

Representante do PPG: Ivan Rogério da Silva

Prof. Dr. Daniel Lucrédio

Prof. Dr. Nelio Alessandro Azevedo Cacho

Prof. Dr. Delano Medeiros Beder

Prof. Dr. Jó Ueyama

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Fabiano Cutigi Ferrari, Daniel Lucrédio, Nelio Alessandro Azevedo Cacho, Delano Medeiros Beder, Jó Ueyama e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Prof. Dr. Fabiano Cutigi Ferrari

(x) Não houve alteração no título ( ) Houve alteração no título. O novo título passa a ser:

Observações:
a) Se o candidato for reprovado por algum dos membros, o preenchimento do parecer é obrigatório.
b) Para gozar dos direitos do título de Mestre ou Doutor em Ciência da Computação, o candidato ainda precisa ter sua dissertação ou tese homologada pelo Conselho de Pós-Graduação da UFSCar.

Agradecimentos

Agradeço imensamente aos meus pais, irmãos e familiares, por todo o apoio incondicional e por serem o alicerce da minha vida.

Minha sincera gratidão à Universidade Federal de São Carlos (UFSCar) e, em especial, ao Departamento de Computação, pela excelente estrutura e ambiente que foram essenciais para este trabalho.

Ao meu orientador, Fabiano Cutigi Ferrari, minha gratidão por sua paciência, sabedoria e por ser um farol em cada etapa da pesquisa, me ensinando a pensar de forma crítica e organizada.

Agradeço o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), cujos auxílios financeiros viabilizaram a execução desta pesquisa.

Por fim, agradeço aos meus amigos, que tornaram esta jornada mais leve e os momentos mais doces.

Resumo

Contexto: Cidades Inteligentes conectam suas infraestruturas física, tecnológica, social e de negócios a fim de melhorar o bem-estar dos cidadãos. As cidades têm sofrido grande crescimento ao longo dos anos, impulsionando a necessidade de Aplicações para Cidades Inteligentes. Devido ao ambiente heterogêneo de tais aplicações, cenários de falha podem nunca ser testados. Assim, é importante o uso de técnicas eficientes de tolerância a defeitos que garantam a confiabilidade de dados utilizados por tais aplicações. **Objetivo:** modelar e avaliar uma abordagem de validação e recuperação de erros em dados de sensores de Aplicações para Cidades Inteligentes, com foco primeiramente na etapa de validação de dados, isto é, na detecção de dados incorretos de sensores de Aplicações para Cidades Inteligentes. **Metodologia:** Para alcançar o objetivo geral, os seguintes passos foram realizados: (i) uma análise do estado da arte a respeito de técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes; (ii) a elaboração de uma abordagem de validação e recuperação de erros adequada para cenários de Cidades Inteligentes; (iii) a seleção e implementação de algoritmos para validação de dados de sensores; (iv) a avaliação dos algoritmos Isolation Forest, Support Vector Machines (SVM) One-Class, e Diversidade por Correlação propostos para detecção de erros. **Resultados e Conclusões**: Como contribuições teóricas, caracterizou-se o estado da arte sobre técnicas de tolerância a defeitos no contexto investigado, e concebeu-se uma abordagem para validar e corrigir dados oriundos de sensores em Aplicações para Cidades Inteligentes. Como contribuições práticas, implementaram-se funcionalidades de software que viabilizam a condução de estudos experimentais de detecção de erros em dados de sensores. Com relação aos experimentos conduzidos, todos os algoritmos apresentaram bons resultados na detecção de erros para o cenário de aplicação avaliado, sendo que a Diversidade por Correlação apresentou o melhor desempenho em menor tempo de execução.

Palavras-chave: Cidades Inteligentes. Aplicações para Cidades Inteligentes. Tolerância a Defeitos. Detecção de Erros.

Abstract

Context: Smart Cities Applications connect their physical, technological, social, and business infrastructures to improve the citizens well-being. Cities have experienced significant growth over the years, propelling the demand for innovative Smart City applications. Due to the heterogeneous environment of such applications, failure scenarios can never be tested. Therefore, it is important to use efficient fault tolerance techniques that ensure the reliability of the data used by such applications. **Objetive:** To model and evaluate an approach for validation and error recovery in sensor data from Smart Cities Applications, with an initial focus on the data validation stage, that is, on detecting incorrect sensor data from Smart Cities Applications. **Methodology:** To achieve the overall objective, the following steps were carried out: (i) an analysis of the state of the art regarding fault tolerance techniques in Smart Cities Applications; (ii) the development of a validation and error recovery approach suitable for Smart Cities scenarios; (iii) the selection and implementation of algorithms for sensor data validation; (iv) the evaluation of the Isolation Forest, One-Class Support Vector Machines (SVM), and Correlation Diversity algorithms proposed for error detection. **Results and Conclusions:** As theoretical contributions, the state of the art on fault tolerance techniques in the investigated context was characterized, and an approach was conceived to validate and correct data from sensors in Smart Cities Applications. As practical contributions, software functionalities were implemented to enable the conduction of experimental studies on error detection in sensor data. Regarding the experiments, all algorithms presented good results in error detection for the evaluated application scenario, with Diversity presenting the best performance in the shortest execution time.

Keywords: Smart Cities. Smart City Applications. Fault Tolerance. Error Detection.

Lista de ilustrações

Lista de tabelas

Lista de siglas

API	*Application Programming Interface*
C	Conferência
cRET	*Collaborative Reliable Event Transport*
D	Diversidade
DE	Defeito Específico
DG	Defeito Genérico
DDoS	*Distributed Denial of Service*
E1	Critério de Exclusão 1
E2	Critério de Exclusão 2
FOCUSeR	*Fog Online Context-Aware Up-to-Date Sensor Ranking*
FE	Falha Específica
FG	Falha Genérica
I1	Critério de Inclusão 1
I2	Critério de Inclusão 2
I3	Critério de Inclusão 3
IoT	*Internet of Things* (Internet das Coisas)
ISO	*International Organization for Standardization*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*
L1	Lote de dados 1
L2	Lote de dados 2
MSIGC	*Mobile Sensing Information Gathering Centers*
MQTT	Message Queuing Telemetry Transport Protocol
P	Periódico
QoI	*Quality of Information*
QP1	Questão de Pesquisa 1
QP2	Questão de Pesquisa 2
QP3	Questão de Pesquisa 3
R	Retentar
REST	*Representational State Transfer*
RNC	Redes Neurais Convolucionais
RNN	Redes Neurais Recorrentes
RSL	Revisão Sistemática da Literatura
RB1	Rodada de Busca 1
RB2	Rodada de Busca 2
RB3	Rodada de Busca 3
RDS	Redes Definidas por Software
RS	Reconfiguração de Sistema
RSW	Redes de Sensores Wireless
RFID	*Radio Frequency Identification*
SA	Sistema Adaptativo
SB	*Snowballing Backward*
SEAL	*SEcure e AgiLe*
SF	*Snowballing Forward*
SOA	*Service-Oriented Architecture*
StArt	*State of the Art through Systematic Review*
SSC	Sistema Sensível ao Contexto
SVM	*Support Vector Machines*
SWIFT	*Smart-based Infrastructural Framework for Smart Transactions*
TIC	Tecnologias da Informação e Comunicação
WS	*Web of Science*
W	Workshop
V2V	*Vehicle-to-Vehicle*
V2X	*Vehicle-to-Everything*

Sumário

Capítulo 1

Introdução

1.1 Contexto e Motivação

Cidades são uma das maiores estruturas que humanos já produziram. Muitas nações como a China, a Índia e outros países emergentes estão sofrendo com o crescimento populacional causado principalmente pela migração de pessoas de áreas rurais para áreas urbanas (HARRISON et al., 2010). Tal migração é motivada pela busca por oportunidades de emprego, pela melhoria na qualidade de vida e por diferentes serviços que as cidades oferecem para seus cidadãos (SÃNCHEZ-CORCUERA et al., 2019). De acordo com o Departamento de Assuntos Econômicos e Sociais das Nações Unidas, na década de 1950, cerca de 30% da população mundial residia em áreas urbanas. De acordo com o mesmo departamento, em 2014, o nível de urbanização atingiu 55%. Para o futuro, a previsão é de que em 2050 cerca de 66% da população do mundo resida em áreas urbanas (United Nations, 2019).

O crescimento das cidades aumenta a pressão sobre sua infraestrutura, exigindo que gestores públicos e privados melhorem a gestão dos recursos da cidade com o objetivo de garantir o bem-estar de seus cidadãos. Assim, surgem as Cidades Inteligentes, que tentam otimizar o uso de seus recursos e o bem-estar urbano. Dentre várias definições encontradas na literatura, uma definição que caracteriza bem Cidades Inteligentes é a apresentada por Harrison et al. (2010), sendo ela: Cidades Inteligentes são cidades que conectam suas infraestruturas física, tecnológica, social e de negócios. O conceito de estrutura física da cidade é estendido para o conceito de infraestrutura virtual, que inclui o uso de sensores, atuadores, dispositivos de comunicação, softwares, entre outros. Neste contexto, surgem as Aplicações para Cidades Inteligentes, que auxiliam gestores e cidadãos a utilizarem de maneira otimizada os recursos da cidade por meio de diferentes tecnologias.

A confiabilidade de Aplicações para Cidades Inteligentes é uma propriedade importante para garantir a crescente aceitação do uso das aplicações por parte dos cidadãos. Tais aplicações são baseadas em contexto, ou seja, dados do ambiente são analisados enquanto as aplicações adaptam seu comportamento de acordo com tal análise. Um falha na aplicação, gerada por um erro, pode causar danos à gestão da cidade e aos cidadãos, dependendo de sua criticidade. Deste modo, o estudo e a aplicação de técnicas de tolerância a defeitos se torna fundamental para garantir a confiabilidade de Aplicações para Cidades Inteligentes em um ambiente de alta variabilidade contextual e tecnológica (PUIU et al., 2016). Além disso, sabe-se que não é possível criar sistemas dessa complexidade que sejam livres de defeitos, sendo teoricamente e praticamente impossível testar exaustivamente um software o suficiente para garantir que o sistema esteja livre de defeitos. A atividade de teste reduz os defeitos e consequentes falhas e, portanto, os riscos de usar esses sistemas, mas não pode garantir que todos os defeitos de software sejam eliminados (MYERS; SANDLER; BADGETT, 2011; AMMANN; OFFUTT, 2017). Os resultados apresentados por alguns estudos secundários também indicam a necessidade de esforços para garantir tolerância a defeitos no contexto de Aplicações para Cidades Inteligentes (SANTANA et al., 2017; NASCIMENTO; OLIVEIRA, 2021).

1.2 Definição do Problema e Justificativa

A fim de compreender o estado da arte a respeito de tolerância a defeitos com suas abordagens e técnicas no cenário de Aplicações para Cidades Inteligentes, foi realizada uma Revisão Sistemática da Literatura (RSL). Com os resultados da revisão, observou-se o problema de garantir a corretude de dados que permeiam uma aplicação, pois além de garantir a disponibilidade da aplicação, é preciso garantir que os dados estejam corretos em aplicações que utilizam diferentes fontes de contexto para tomada de decisão. Diante deste cenário, um dos principais problemas identificados sumariza-se em como garantir a confiabilidade de dados que trafegam em Aplicações para Cidades Inteligentes, de modo que, estejam atualizados e corretos, em um ambiente que recebe dados de fontes heterogêneas de informação (PUIU et al., 2016; YUAN et al., 2021; KHAN; KIM; PARK, 2022; COSTA; NASSAR; DANTAS, 2022).

Para conceber a abordagem para solucionar o problema identificado analisaram-se as características das técnicas utilizadas nos trabalhos relacionados, observando os desafios de cada uma. Analisando-se as técnicas de tolerância a defeitos dedicadas à análise de dados para fins de detecção de erros, notou-se que a maioria dos estudos explora o uso de técnicas simples como *timeouts* (isto é, técnicas baseadas em tempo de resposta). Técnicas de detecção de erros baseadas em *timeouts* possuem o desafio de entregar o dado em um tempo limite, mas não garantem que o dado entregue para a aplicação esteja correto, ou seja, que o valor esteja de acordo (isto é, preciso) com o contexto lido pelo sensor.

Outros trabalhos apresentam técnicas baseadas em descrição de sensores (PUIU et al., 2016) e baseadas em análise de limites de valores pré-definidos (YUAN et al., 2021). Contudo, tais abordagens apresentam o desafio de requererem a atenção de especialistas para a definição das descrições e dos limites de valores pré-definidos para diferentes contextos, tornando a abordagem inflexível diante de mudanças dinâmicas.

Técnicas baseadas em análises de séries temporais também foram utilizadas. Khan, Kim e Park (2022) apresentaram uma abordagem para a detecção de anomalias em sequências de dados de sensores, por meio da análise valores menores, iguais a zero, picos abruptos e diminuições rápidas nos dados. Contudo, tais análises podem dificultar a detecção de erros com mudanças sutis que não caracterizam padrões abruptos.

Outra técnica de detecção de anomalias em séries temporais também foi utilizada. O trabalho de Costa, Nassar e Dantas (2022) apresentou uma abordagem para detecção de anomalias por meio da técnica Matriz de Perfil. De acordo com os autores, esta abordagem se mostra promissora para cenários que exigem baixa latência. Embora os autores mencionem resultados promissores, tal técnica apresenta um alto custo computacional para séries temporais muito longas.

Diante da criticidade de cenários de falha em Aplicações para Cidades Inteligentes e de poucos estudos que buscam estratégias para validar dados de sensores, percebe-se que há campo de exploração para o estudo de novos algoritmos para a validação de dados. Nessa linha, inicialmente objetiva-se maximizar a detecção de dados incorretos e, em seguida, diminuir o tempo de processamento. Dentre as técnicas observadas, técnicas baseadas em análise de séries temporais se destacaram por permitir a detecção automática de erros com base nos próprios dados.

1.3 Objetivos Geral e Específicos

Diante do problema apresentado e das limitações das abordagens de detecção de erros em dados de sensores de Aplicações para Cidades Inteligentes, este trabalho tem como *objetivo geral* propor uma solução voltada para Cidades Inteligentes com foco em tolerância a defeitos para garantir a confiabilidade de dados dessas aplicações, uma vez que falhas em tais aplicações podem causar prejuízos aos usuários que as utilizam, assim como aos gestores da cidade.

Para alcançar o objetivo geral deste trabalho, os seguintes *objetivos específicos* foram estabelecidos:

1. Elaborar uma abordagem para validação e recuperação de erros em dados de sensores em Aplicações para Cidades Inteligentes.

2. Investigar e selecionar algoritmos para validação de dados de sensores.

3. Avaliar os algoritmos selecionados para validação de dados de sensores.

1.4 Metodologia

Considerando os objetivos estabelecidos, a metodologia utilizada neste trabalho foi realizada de acordo com as seguintes etapas: (i) avaliação do estado da arte a respeito de tolerância a defeitos em Aplicações para Cidades Inteligentes; (ii) definição de uma abordagem de tolerância a defeitos, ou seja, uma abordagem de validação e recuperação de erros em dados de Aplicações para Cidades Inteligentes; (iii) a seleção e implementação de algoritmos para validação de dados de sensores; e (iv) a avaliação dos algoritmos propostos. Para atender à etapa (i), primeiramente foi conduzida uma (RSL) a fim de reunir evidências sobre as técnicas de tolerância a defeitos propostas ou aplicadas em Aplicações para Cidades Inteligentes. Para atender à etapa (ii), foi realizada uma análise da literatura e a observação de soluções promissoras para o contexto de Cidades Inteligentes. Para atender à etapa (iii), foi realizada uma análise de possíveis algoritmos utilizados para validação (observando-se o estado da arte elencado, que consiste em um estudo secundário que menciona possibilidades de algoritmos de validação), e a busca por características que atendessem às necessidades dessas aplicações. Por fim, para atender à etapa (iv) foi realizado um estudo experimental com simulações de dados de Aplicações para Cidades Inteligentes com o objetivo de avaliar os algoritmos propostos.

1.5 Organização do Trabalho

Neste capítulo foram apresentados o contexto geral, a definição do problema, a justificativa, os objetivos geral e específicos deste trabalho, e uma breve descrição da metodologia adotada para atindir os objetivos estabelecidos. Os próximos capítulos estão organizados da seguinte forma:

- No Capítulo 2 apresentam-se os principais conceitos relacionados a Cidades Inteligentes, exemplos de cidades inteligentes no Brasil e no mundo, características de aplicações para cidades inteligentes e conceitos sobre Dependabilidade e Tolerância a Defeitos;

- Capítulo 3 são apresentadas as duas revisões da literatura conduzidas. A primeira, apresentada em mais detalhes, sobre técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes e a segunda sobre técnicas de tolerância a defeitos em SAs e SSCs, discutida brevemente.

- No Capítulo 4 é apresentada a proposta de trabalho elaborada nesta tese bem como a justificativa da proposta.

- No Capítulo 5 é descrito o estudo experimental realizado para avaliar a proposta.

- No Capítulo 6 são discutidas as conclusões da tese e as alternativas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Considerações Iniciais

As cidades inteligentes são caracterizadas pelo uso de tecnologias avançadas para melhorar a qualidade de vida dos seus habitantes, integração de diferentes serviços urbanos e otimização da gestão de seus recursos. Nesse contexto, as Aplicações para Cidades Inteligentes são fundamentais para impulsionar o desenvolvimento de tais cidades. É necessário garantir soluções eficientes de tolerância a defeitos [1] em Aplicações para Cidades Inteligentes, principalmente em aplicações cujas falhas podem ser críticas e afetar negativamente seus usuários. Deste modo, o entendimento desses tópicos torna-se fundamental.

Nesse contexto, neste capítulo são apresentados os principais conceitos que embasam esta proposta de doutorado. Na Seção 2.2 são apresentadas diferentes definições para Cidades Inteligentes e alguns exemplos de Cidades Inteligentes no Brasil e no mundo. Na Seção 2.3 são apresentadas características de Aplicações para Cidades Inteligentes e a relação que pode ser estabelecida entre essas aplicações e Sistemas Adaptativos (SAs) bem como Sistemas Sensíveis ao Contexto (SSCs). Por fim, na Seção 2.5 são apresentados os conceitos sobre Dependabilidade e Tolerância a Defeitos.

2.2 Cidades e Cidades Inteligentes

Cidades são uma das maiores estruturas sociais, econômicas, culturais e defensivas que humanos já produziram. Muitas nações com a China, a Índia e outros países emergentes estão sofrendo com o crescimento populacional causado principalmente pela migração

[1] O objetivo da tolerância a defeitos é aplicar técnicas que evitem que defeitos e consequentes erros ativos no sistema levem à falha do sistema (LEE; ANDERSON, 1990).

de pessoas de áreas rurais para áreas urbanas. Tal crescimento aumenta a pressão e a sobrecarga sobre a estrutura física e os serviços fornecidos pela cidade como, por exemplo, a infraestrutura de distribuição de eletricidade e água, estradas, segurança pública etc. (HARRISON et al., 2010). A migração de pessoas de áreas rurais para áreas urbanas é motivada pela busca por oportunidades de emprego, melhoria na qualidade de vida e por diferentes serviços que as cidades oferecem a seus residentes. Mais ofertas de trabalho, qualidade na educação, saúde e serviços de transporte público são exemplos de fatores que aumentam essa busca (SÃNCHEZ-CORCUERA et al., 2019).

De acordo com Harrison et al. (2010), uma cidade se caracteriza como uma Cidade Inteligente quando existe uma conexão entre a infraestrutura física, a infraestrutura de tecnologias da informação, e a infraestrutura social e de negócios da cidade. O conceito de estrutura física da cidade é estendido para o conceito de infraestrutura virtual, que inclui o uso de sensores, transmissões de banda larga, aplicações de software, entre outros. Tal infraestrutura permite que dados da cidade sejam coletados e analisados, de modo que tomadores de decisões que administram a cidade usem estes dados e possam tomar decisões adequadas que gerem um impacto positivo na vida dos cidadãos.

O conceito de Cidades Inteligentes tem sido explorado desde 2005 por meio do avanço de diferentes tipos de tecnologias de informação e comunicação, internet das coisas, *big data*, mineração e fusão de dados, entre outros. Exemplos de uso de tais tecnologias aparecem nos domínios de construção civil, transporte e mobilidade urbana, distribuição de água e energia elétrica, saúde, habitação, segurança pública, gestão ambiental, e setor industrial e agrícola. A integração de tais tecnologias permite que gestores municipais utilizem diferentes tipos de informações para aprimorar o planejamento e a gestão dos recursos da cidade. Várias cidades ao redor do mundo já têm aproveitado o uso de tecnologias para melhorar o conforto, segurança, mobilidade, saúde e o bem-estar de seus cidadãos (LAU et al., 2018).

Uma definição para Cidades Inteligentes, conforme apresentada por Harrison et al. (2010) e por Dameri (2013), baseia-se no uso de Tecnologias da Informação e Comunicação (TICs) como um meio de aprimorar os serviços e a infraestrutura da cidade. Observa-se, porém, que na literatura encontram-se outras definições para Cidades Inteligentes. Algumas delas estão mais apoiadas no uso de tecnologia, enquanto outras estão mais relacionadas ao meio-ambiente e sustentabilidade. A seguir, são descritas algumas definições mencionadas por diferentes estudos.

Para Washburn e Sindhu (2009) uma Cidade Inteligente utiliza tecnologias da informação para tornar componentes e serviços essenciais de uma cidade como, por exemplo, serviços de administração municipal, educação, saúde, segurança pública e transporte, mais inteligentes, interconectados e eficientes.

Para Caragliu, Bo e Nijkamp (2011), uma cidade é inteligente quando os investimentos em capital humano, social e na infraestrutura de comunicação impulsionam o crescimento

econômico e sustentável da cidade, além de fornecer alta qualidade de vida aos cidadãos. Para Dameri (2013), uma Cidade Inteligente é uma área geográfica na qual a cooperação de tecnologias da informação e comunicação, logística e infraestruturas de produção de energia criam benefícios para os cidadãos como, por exemplo, aumentando seu bem-estar e proporcionando inclusão social, desenvolvimento sustentável e inteligente. Uma Cidade Inteligente é uma cidade digital, pois usa tecnologias da informação e comunicação tanto para processamento de dados quanto para compartilhamento de informações, mas também para apoiar a comunicação; é sustentável, pois usa a tecnologia para reduzir as emissões de CO2, produzir energia eficiente, rumando a tornar-se uma cidade verde; é uma cidade que usa a tecnologia para melhorar a eficiência e eficácia das suas infraestruturas e serviços visando a melhoria na qualidade do espaço urbano, mobilidade, transportes públicos, logística; é uma cidade que fornece qualidade de vida para os cidadãos e cria possibilidades para os negócios e para as pessoas. A tecnologia é umas das ferramentas utilizadas para alcançar esses objetivos, assim como cultura, o clima, a história e monumentos que são considerados fatores importantes para uma Cidade Inteligente.

De acordo com Giffinger et al. (2007) uma Cidade Inteligente é construída pela combinação de seis dimensões: economia inteligente (*smart economy*), pessoas inteligentes (*smart people*), governança inteligente (*smart governance*), mobilidade inteligente (*smart mobility*), ambiente inteligente (*smart environment*), e vida inteligente (*smart living*). Tais dimensões podem ser descritas como:

- Economia inteligente: fazem parte desta dimensão projetos relacionados ao crescimento e desenvolvimento econômico da cidade que atraem investimentos e criam oportunidades de trabalho.

- Pessoas inteligentes: fazem parte desta dimensão projetos que visam o desenvolvimento da população e envolvem a melhoria de parâmetros sociais, tais como educação, taxa de emprego e renda;

- Governança inteligente: constituída por projetos que visam facilitar a administração da cidade, permitir a participação política dos cidadãos em decisões estratégicas, e facilitar o uso de serviços públicos;

- Mobilidade inteligente: inclui projetos que envolvem o uso de sistemas de transporte modernos e sustentáveis para a melhoria do fluxo de pessoas na cidade, e o uso de sistemas para o monitoramento de estradas e estações de metrô;

- Ambiente inteligente: envolve projetos que visam a melhoria de indicadores de sustentabilidade como, por exemplo, a redução da poluição do ar e água, o aperfeiçoamento da distribuição bem como o uso inteligente de recursos da cidade como água e eletricidade, e o aprimoramento de serviços como coleta de lixo e reciclagem; e

- ❏ Vida inteligente: constituída por projetos que visam melhorar a qualidade de vida da população, aprimorando serviços relacionados a atividades culturais, esportivas e de entretenimento e segurança.

Além das dimensões citadas por Giffinger et al. (2007), outros indicadores e esquemas têm sido propostos na literatura para monitorar e avaliar o progresso de Cidades Inteligentes. Sharifi (2020) analisou 38 esquemas de classificação para avaliar a inteligência (*smartness*) de bairros, comunidades, cidades e regiões urbanas. O autor menciona que embora não haja uma definição universal para o conceito de Cidades Inteligentes, há um consenso de que elas utilizam soluções de Tecnologia da Informação e Comunicação (TICs) em vários domínios (socioeconômico, institucional e ambiental) para aumentar a qualidade de vida dos cidadãos e a sustentabilidade e a resiliência da cidade, mantendo sua capacidade competitiva em uma rede de cidades interconectadas.

Sãnchez-Corcuera et al. (2019) discutem diferentes definições para Cidades Inteligentes já relatadas na literatura e revisam taxonomias de tipos domínios de Aplicações para Cidades Inteligentes. Os autores descrevem quatro possíveis definições para Cidades Inteligentes de acordo com a taxonomia proposta por Yin et al. (2015). Tais definições são:

1. Definições baseadas em tecnologia: são centradas na importância de tecnologia, ou seja, o uso de tecnologias da informação e comunicação é suficiente para tornar uma cidade uma Cidade Inteligente;

2. Definições baseadas em domínio: são focadas em explicar Cidades Inteligentes do ponto de vista de domínios como educação, governança e cuidados com a saúde;

3. Definições baseadas na integração de sistemas: tecnologias podem ser utilizadas em cada setor de uma cidade, mas se o sistema não está integrado como um todo, a cidade não pode ser considerada inteligente, o foco é na integração de todos os elementos da cidade; e

4. Definições com foco em dados: definições com foco na transmissão e no uso de dados por meio de protocolos de comunicação (embora essa definição também possa ser categorizada como uma definição baseada em tecnologia, o foco é a coleta de dados e o fornecimento de serviços que usem esses dados).

O índice *The Cities of the Future Index*[2] de 2021 classifica as cidades mais inteligentes do mundo que estão liderando o caminho para abraçar grandes avanços tecnológicos e um caminho de sustentabilidade. A lista foi criada pelo grupo EasyPark de modo que cidades ao redor do mundo foram classificadas de acordo com 14 indicadores e, ao final, foi gerada uma lista com 150 principais cidades com base nas pontuações totais. Há também

[2] <https://easyparkgroup.com/studies/cities-of-the-future/en/> - acessado em novembro de 2024.

uma classificação usada para ranquear as cidades mais inteligentes da Europa desenvolvida pela Universidade de Tecnologia de Viena. Sua versão mais atual é o Europeansmartcities 4.0 de 2015[3] que classifica as cidades mais inteligentes de porte médio da Europa com número de habitantes entre 300.000 e 1 milhão. O International Institute for Management Development (IMD),[4] em parceria com a World Smart Sustainable Cities Organization (WeGO), mantém outro índice, chamado de *IMD Smart Cities Index*,[5] que teve sua última edição liberada em 2024. A classificação lista 142 cidades mais inteligentes no mundo que são classificadas de acordo com 15 indicadores.

2.2.1 Cidades Inteligentes no Brasil

No Brasil, existem diversas iniciativas para transformar cidades em Cidades Inteligentes. Um estudo que tem sido mantido pelas organizações Necta e Urban Systems desde 2015 mapeia, atualmente, 656 municípios do Brasil com mais de 50 mil habitantes, com o objetivo de mostrar quais são as cidades mais inteligentes do país de acordo com 74 indicadores que estão distribuídos nos seguintes onze eixos temáticos: Mobilidade, Urbanismo, Meio Ambiente, Energia, Tecnologia e Inovação, Economia, Educação, Saúde, Segurança, Empreendedorismo e Governança. Com base em tal estudo, foi criada uma listagem (ranking) denominada *Connected Smart Cities*.[6] Na Tabela 1 listam-se as 10 cidades mais inteligentes do Brasil de acordo com o ranking criado. Na tabela, são apresentados, nessa ordem, a posição da cidade, o estado, o nome da cidade, a região de localização, e seu porte em termos de número de habitantes. A listagem, disponível em plataforma digital,[7] também permite a consulta de acordo com filtros, tais como o porte da cidade, o estado, a região e eixo temático. Para este último é possível selecionar os eixos temáticos de interesse. Para calcular a posição de cada cidade, foi utilizado para cada eixo o respectivo conjunto de indicadores. Por exemplo, para o eixo de Mobilidade, foram calculados os seguintes indicadores: proporção de automóveis por habitante, idade média das frotas de veículos em anos, relação entre a quantidade de ônibus e automóveis, disponibilidade de outros modos de transporte coletivo, quilometragem de ciclovias, transporte rodoviário (quantidade de conexões interestaduais), quantidade de aeroportos em um raio de 100 km, veículos de baixa emissão (porcentagem do total da frota), existência de bilhete eletrônico do transporte público, presença de semáforos inteligentes, e taxa de mortes em acidente de trânsito (quantidade de mortes a cada 100 mil habitantes).

De acordo com Talebkhah et al. (2021) uma Cidade Inteligente tem como objetivo melhorar a eficácia e a utilidade de seus sistemas e serviços. Dentre as suas principais

[3] <https://www.smart-cities.eu> - acessado em novembro de 22024.
[4] <https://www.imd.org/smart-city-observatory/home/> - acessado em novembro de 2024.
[5] <https://imd.widen.net/s/q7flvgtvbs/20240412-smartcityindex-2024-full-report_4> - acessado em novembro de 2024.
[6] <https://ranking.connectedsmartcities.com.br/> - acessado em novembro de 2024.
[7] <https://bit.ly/plataformarcsc24> - acessado em novembro de 2024.

características estão:

- ❏ Uma infraestrutura sólida e completa, assegurando acesso protegido e livre aos recursos.
- ❏ Um projeto voltado para as necessidades dos cidadãos, dando prioridade às suas necessidades e interações.
- ❏ A utilização de grandes quantidades de dados, tanto públicos quanto privados, oriundos de aparelhos móveis e vestíveis, que podem ser guardados, acessados, compartilhados e classificados, possibilitando que os cidadãos tenham acesso a informações em qualquer lugar quando necessário.
- ❏ Aplicações que mesclam funções analíticas e integradas para simplificar o processo decisório.
- ❏ Uma rede física e tecnológica sofisticada que possibilita a transferência eficaz de grandes quantidades de dados variados e o apoio a serviços e aplicações mais intrincados e dispersos.

A seguir, são discutidos alguns exemplos concretos de iniciativas em cidades que se encontram entre as 10 ranqueadas como mais inteligentes do Brasil, segundo o ranking *Connected Smart Cities*. Em particular, apresentam-se exemplos referentes às cidades de Florianópolis, São Paulo e Curitiba.

Tabela 1 – Classificação das cidades mais inteligentes do Brasil de acordo com o ranking *Connected Smart Cities* - Edição 2024.

Posição	**Estado**	**Município**	**Região**	**Porte**
1°	Santa Catarina	Florianópolis	Sul	Mais de 500 mil
2°	Espírito Santo	Vitória	Sudeste	De 100 a 500 mil
3°	São Paulo	São Paulo	Sudeste	Mais de 500 mil
4°	Paraná	Curitiba	Sul	Mais de 500 mil
5°	Rio de Janeiro	Niterói	Sudeste	De 100 a 500 mil
6°	Santa Catarina	Balneáreo Camburiú	Sul	De 100 a 500 mil
7°	São Paulo	São Caetano do Sul	Sudeste	De 100 a 500 mil
8°	Minas Gerais	Belo Horizonte	Sudeste	Mais de 500 mil
9°	São Paulo	Barueri	Sudeste	De 100 a 500 mil
10°	Bahia	Salvador	Nordeste	Mais de 500 mil

Em **Florianópolis**, o Serviço Nacional de Aprendizagem Comercial (Senac) e a Administração Regional de Santa Catarina elaboraram um estudo intitulado “Smart Floripa 2030: Transformando Florianópolis numa cidade inteligente de inovação”, que é um plano para torná-la uma cidade ainda mais inteligente e sustentável. Na classificação por eixos do ranking *Connected Smart Cities* - Edição 2024, a cidade obteve as seguintes posições: Segurança: 3^a posição, Tecnologia e Inovação: 5^a posição, Urbanismo: 22^a posição, Economia: 7^a posição, Educação: 25^a posição, Empreendedorismo: 8^a posição, Governança:

77^a posição, Meio Ambiente: 67^a posição, Mobilidade: 3^a posição, e Saúde: 4^a posição (ressalta-se que Edição 2024 do ranking não apresentou um recorte relacionado ao eixo Energia). Por exemplo, com relação a mobilidade urbana, eixo na qual a cidade de Florianópolis se destaca, o aplicativo Floripa no Ponto[8] é utilizado para obter informações a respeito de rotas e horários de ônibus em tempo real.

Na classificação por eixos do ranking *Connected Smart Cities* - Edição 2024, a cidade de **São Paulo** obteve as seguintes posições: Segurança: 5^a posição, Tecnologia e Inovação: 9^a posição, Urbanismo: 4^a posição, Economia: 4^a posição, Educação: 50^a posição, Empreendedorismo: 100+ (fora das 100 primeiras posições), Governança: 27^a posição, Meio Ambiente: 48^a posição, Mobilidade: 1^a posição, e Saúde: 51^a posição (novamente, ressalta-se que Edição 2024 do ranking não apresentou um recorte relacionado ao eixo Energia). Algumas iniciativas que compõem o cenário de São Paulo como uma Cidade Inteligente são listadas a seguir. A cidade mantém o Portal de Dados Abertos da Prefeitura de São Paulo[9] que reúne conjuntos de dados de todas as secretarias, subprefeituras e empresas públicas municipais, dessa forma fornecendo dados demográficos da cidade, localização de paradas de ônibus e pontos de alagamento, dados de orçamento e gestão da cidade, etc. Outro exemplo é o GeoSampa,[10] um mapa digital da cidade de São Paulo que provê dados georreferenciados, possibilitando consultas de informações sobre a rede de transporte público, vagas de estacionamento, bibliotecas, escolas, parques, etc. A cidade possui a API Olho Vivo,[11] que fornece informações em tempo real do monitoramento da frota de ônibus da cidade. Com a API é possível criar aplicações que facilitem a mobilidade de cidadãos que utilizam ônibus. A startup Scipopulis desenvolveu um painel web de monitoramento de ônibus, o Painel Trancity,[12] que atualmente é utilizado por gestores da cidade de São Paulo para administrar de forma inteligente suas frotas de ônibus. Exemplos de dados apresentados pelo painel são a velocidade e a posição dos ônibus nas ruas, acidentes, a quantidade de veículos que passam em determinado percurso, etc. No tocante à gestão do transporte público, o painel auxilia a identificar pontos de melhoria como, por exemplo, trajetos que precisem de maior volume de ônibus, horários que necessitam de maior e menor volume de frota, e pontos de interrupção de fluxo de veículos.

Em **Curitiba** o aplicativo Saúde Já, disponível para as plataformas Android, IOs e também uma versão Web[13] permite que usuários agendem o atendimento a uma unidade municipal de saúde. O Central 156[14] é um aplicativo que permite a comunicação entre

[8] <https://www.pmf.sc.gov.br/entidades/defesa/index.php?pagina=notpagina&noti=25747> - acessado em novembro de 2024.

[9] <http://dados.prefeitura.sp.gov.br/> - acessado em novembro de 2024.

[10] <http://geosampa.prefeitura.sp.gov.br/PaginasPublicas/_SBC.aspx> - acessado em novembro de 2024.

[11] <https://www.sptrans.com.br/desenvolvedores/api-do-olho-vivo-guia-de-referencia/documentacao-api/> - acessado em novembro de 2024.

[12] <https://www.scipopulis.com/> - acessado em novembro de 2024.

[13] <https://saudeja.curitiba.pr.gov.br/> - acessado em novembro de 2024.

[14] <https://156.curitiba.pr.gov.br/> - acessado em novembro de 2024.

cidadãos e a Prefeitura de Curitiba. Por meio do aplicativo, é possível verificar itinerários e acompanhar o deslocamento de veículos em tempo real. Na classificação por eixos do ranking *Connected Smart Cities* - Edição 2024, a cidade obteve as seguintes posições: Segurança: 8^a posição, Tecnologia e Inovação: 1^a posição, Urbanismo: 42^a posição, Economia: 25^a posição, Educação: 26^a posição, Empreendedorismo: 4^a posição, Governança: 8^a posição, Meio Ambiente: 10^a posição, Mobilidade: 37^a posição, e Saúde: 8^a posição.

2.2.2 Cidades Inteligentes no Mundo

Na Tabela 2 é apresentada uma lista das dez cidades mais inteligentes do mundo de acordo com o *IMD Smart City Index* - Edição 2024, mencionado na seção anterior. Em **Zurique**, que é a primeira cidade na classificação, algumas iniciativas têm sido realizadas a fim de tornar a cidade uma cidade inteligente. Um dos projetos de exemplo na cidade são os veículos Pikmi que podem ser reservados por smartfone e são uma opção de transporte coletivo sob demanda. Solicitações de viagem com destinos semelhantes são agrupadas em um mesmo veículo de modo a otimizar o trajeto. Zurique também conta com o *Smart City Lab*, um espaço para funcionários da administração municipal, empresas, universidades e start-ups discutirem métodos de inovação e abordagens criativas relacionadas a Cidades Inteligentes. Além disso, a plataforma online *Housekeeping App*[15] fornece a opção para que cidadãos possam reportar danos e defeitos identificados em qualquer ponto da infraestrutura da cidade.

Uma cidade de destaque que tem aparecido nas posições superiores do *IMD Smart City Index* é a cidade de Singapura (5^a colocada em 2024). A cidade possui o *Smart Nation Singapure*,[16] que consiste de projetos estratégicos para a transformação da cidade de Singapura por meio de tecnologia.

Tabela 2 – Classificação das cidades mais inteligentes no mundo de acordo com *IMD Smart City Index 2024*.

Posição	**Cidade**	**País**
1^a	Zurique	Suiça
2^a	Oslo	Noruega
3^a	Camberra	Austrália
4^a	Genebra	Suíça
5^a	Singapura	Singapura
6^a	Copenhague	Dinamarca
7^a	Lausana	Suiça
8^a	Londres	Reino Unido
9^a	Helsinque	Finlândia
10^a	Abu Dabi	Emirados Árabes Unidos

[15] <https://www.zueriwieneu.ch/> - acessado em novembro de 2024.
[16] <https://www.smartnation.gov.sg/> - acessado em novembro de 2024.

Outra cidade de destaque que tem aparecido nas posições superiores do *IMD Smart City Index* é a cidade de Copenhague (6^a colocada em 2024). Em **Copenhague**, o *Copenhaguen Solutions Lab*[17] apoia o desenvolvimento da cidade por meio de testes e implementação de soluções inteligentes que atendem às necessidades da cidade e de seus cidadãos. Laboratórios urbanos possibilitam a realização de experimentos sistemáticos e de larga escala. Atualmente, estão disponíveis em diversos laboratórios na cidade como, por exemplo, o *AI4Cities*,[18] que foca no uso de inteligência artificial para beneficiar diferentes cidades da Europa (Amsterdã, Helsinque, Copenhague, Paris, Stavanger e Talin; o *Smart Cities Accelerator+*,[19] com foco em novos sistemas de energia inteligente a fim de reduzir a emissão de carbono; e o *Viadukten*,[20] que consiste em uma oficina que combina artesanato com o uso de tecnologias, *Design Thinking*, Prototipagem, etc.

2.3 Sistemas Adaptativos, Sistemas Sensíveis ao Contexto e Aplicações para Cidades Inteligentes

Sistemas Adaptativos (SAs) são sistemas aptos a automaticamente adaptarem seu comportamento em resposta ao ambiente (Oreizy et al., 1999; KEPHART; CHESS, 2003). De acordo com Krupitzer et al. (2015), a razão para adaptação pode ser uma mudança em um ou vários elementos do sistema, tais como: (i) uma mudança em recursos técnicos, tais como pela ocorrência defeitos de software ou hardware, (ii) uma mudança em um estado de uma variável de contexto, ou (iii) uma mudança em relação a preferências de usuário. As propriedades gerais de um SA podem ser sumarizadas de acordo com os quatro seguintes objetivos ou propriedades, conhecidas como propriedades self*:

- Autoconfiguração (do inglês, *self-configuration*): capacidade de um sistema modificar sua configuração dinamicamente em resposta a alterações de contexto, podendo inserir ou remover componentes do sistema.

- Autocura (do inglês, *self-healing*): capacidade de um sistema detectar e diagnosticar problemas que causam interrupções no sistema. Tais problemas podem ser causados por falhas de hardware, falhas de software, erros em bases de dados etc. A partir do diagnóstico, o sistema deve ser capaz de se recuperar da falha retornando a seu estado normal de operação.

[17] <https://cphsolutionslab.dk/> - acessado em novembro de 2024.
[18] <https://ai4cities.eu/cities/copenhagen> - acessado em novembro de 2024.
[19] <https://cphsolutionslab.dk/en/projekter/samarbejder/smart-cities-accelerator> - acessado em novembro de 2024.
[20] <https://cphsolutionslab.dk/en/projekter/samarbejder/underbroen> - acessado em novembro de 2024.

- Auto-otimização (do inglês, *self-optimisation*): capacidade de um sistema de otimizar o uso de recursos¸ ajustando-se dinamicamente para melhorar o desempenho ou a qualidade de seu serviço.

- Autoproteção (do inglês, *self-protection*): capacidade de um sistema de antecipar, detectar e proteger-se contra ameaças, tais como ataques maliciosos ou até mesmo usuários que podem fazer alguma mudança não-intencional no software como, por exemplo, apagar algum arquivo importante.

SAs são compostos por dois subsistemas que se relacionam: o subsistema gerenciado e o subsistema gerenciador (LALANDA; MCCANN; DIACONESCU, 2013; SHAW, 1995). O subsistema gerenciado implementa o domínio de aplicação. O subsistema gerenciador monitora, analisa e planeja mudanças sobre o sistema gerenciado disparando adaptações quando necessário. O subsistema gerenciador é composto por *loops* de controle (GARLAN et al., 2004; SHAW, 1995; Weyns et al., 2013). O MAPE-K (*Monitor, Analyzer, Planner, Executor - Knowledge-based*) é um modelo conceitual de *loop* de controle proposto pela IBM (2006) e serve como base para a estruturação de SAs especificando seus elementos conceituais. O gerenciador autônomo que representa o *loop* de controle MAPE-K é composto por quatro elementos básicos: monitores, analisadores, planejadores e executores. Os monitores têm o objetivo de monitorar e coletar dados provenientes do sistema gerenciado. Os analisadores permitem que o gerenciador autônomo aprenda sobre o ambiente e ajude a predizer situações futuras. Os planejadores, por sua vez, criam as ações a serem tomadas para atingir o objetivo. Os executores, por fim, aplicam as adaptações do que foi planejado. Tem-se ainda um componente de conhecimento (K), que diz respeito à coleta dos dados que possibilita decisões baseadas nesse conhecimento.

Sistemas Sensíveis ao Contexto (SSCs), por sua vez, são aptos a coletarem dados contextuais a fim de saber qual é o estado do sistema em execução (WANG; CHAN; TSE, 2014). Com base nos dados de contexto coletados, o sistema é apto a tomar decisões de acordo com o objetivo dos usuários. Esses dados de contexto coletados representam o contexto em que o sistema está inserido e são chamados de variáveis de contexto (LU; CHAN; TSE, 2006). Conforme mencionado em Siqueira et al. (2021), SAs e SSCs compartilham algumas características. Por exemplo, ambos compartilham a capacidade de monitorar seu ambiente, avaliar e adaptar seu comportamento a fim de alcançar objetivos.

Assim como em SAs e em SSCs, é possível observar que em Aplicações para Cidades Inteligentes também faz-se uso de sensores com o objetivo de monitorar e coletar dados provenientes do ambiente. Essa capacidade de coleta de dados via sensoriamento e a conectividade com a internet permitem que Aplicações para Cidades Inteligentes cubram um amplo espaço de monitoramento e coleta de dados na cidade. De acordo com a RSL conduzida por Rocha et al. (2022), que avaliou características de SSCs suportados pelas infraestruturas de Cidades Inteligentes, as aplicações relatadas nos estudos analisados in-

tegram sensores implantados na cidade para adquirir atributos de contexto (por exemplo dos cidadãos: localização, velocidade, status de atividades, pontos de interesse, diferentes tipos de emoções). Uma vez que os objetivos e necessidades dos usuários de Aplicações para Cidades Inteligentes são dependentes de tempo e espaço, mecanismos para análise de contexto (*context-awareness*) são necessários para coletar e processar informações contextuais, permitindo que as aplicações utilizem de forma inteligente as condições de seus usuários e ambientes situados por eles (ROCHA et al., 2022).

Cidades com grande densidade populacional elevam a pressão sobre energia, água, construções, áreas públicas e transporte. Assim, é importante buscar soluções que sejam eficazes e viáveis para o desenvolvimento econômico da cidade e da sociedade em geral. A Internet das Coisas (IoT) e a computação em nuvem representam dois elementos cruciais para a conectividade de recursos em Cidades Inteligentes. Dados públicos são elementos que se destacam, servindo como alicerce para a tomada de decisões em tempo real a respeito de situações da cidade. Também são desenvolvidas aplicações que possibilitam o acesso a conjuntos de dados e dados públicos abertos (KAUR; MAHESHWARI, 2016).

As vantagens de utilização de diferentes tecnologias e da IoT em uma Cidades Inteligentes são significativas. Soluções que utilizam dispositivos como medidores inteligentes de energia, dispositivos de segurança, dispositivos inteligentes para saúde e vida doméstica, entre outros, proporcionam comodidade e aprimoram a qualidade de vida de várias comunidades e indivíduos. A ideia de IoT consiste na combinação de componentes como sensores, atuadores, utilização de interfaces de comunicação e da capacidade de processamento na Internet. Elementos do dia a dia presentes em casas, como geladeira, janela, aquecedor, interruptor, máquina de lavar, entre outros, podem ser facilmente acessíveis, gerenciáveis e se comunicarem por meio de protocolos baseados na Internet, como IPv6, UDP/TCP e HTTP. O modelo IoT requer a união de uma vasta gama de dispositivos que se conectam à Internet através de diversos protocolos de rede (KAUR; MAHESHWARI, 2016).

Com a aplicação da tecnologia de Big Data em Cidades Inteligentes é possível armazenar dados de maneira eficaz e transformá-los em conhecimento para a criação de serviços urbanos inteligentes. Portanto, o uso de ferramentas e métodos sofisticados pode conduzir a uma análise de dados altamente eficiente e eficaz torna-se importante. Tais análises têm o potencial de fomentar a colaboração entre as organizações que prestam serviços em diversas áreas da cidade, além de aprimorar a satisfação do cliente e as possibilidades comerciais, o que gera grande interesse por parte dos interessados (TALEBKHAH et al., 2021).

Talebkhah et al. (2021) apresentaram um framework conceitual para o desenvolvimento de Cidades Inteligentes composto pelas quatro camadas descritas a seguir:

- Camada de Percepção: essa camada tem a função de coletar dados, é formada por redes de sensores sem fio (*Wireless Sensor Networks* (WSNs)), sendo responsável por

reunir diferentes tipos de dados. Sua rede de sensores abrange uma grande variedade de capacidades e inclui diversos dispositivos de sensoriamento, como sensores RFID e GPS, tecnologias como Bluetooth e Zigbee, além de atuadores e câmeras.

- Camada de Comunicação (Transferência de Dados): tal camada engloba todas as opções tecnológicas de rede para permitir a comunicação entre elementos físicos e as aplicações. Ela abrange diversos tipos de tecnologias, como redes com fio, sem fio e satélite. As tecnologias de curto alcance, como Zwave, Bluetooth, M2M, Zigbee e RFID, são combinadas com aquelas de maior alcance, como redes de longa distância de baixa potência, 5G, 4G e 3G.

- Camada de Gerenciamento de Dados: pode ser vista como o cérebro da Cidades Inteligentes. Essa camada é responsável por tarefas como manipulação, organização, limpeza, análise, armazenamento de dados e tomada de decisões baseadas em dados. Uma gestão eficiente dos dados é essencial para garantir a sustentabilidade de uma Cidades Inteligentes, pois suas funções estão intimamente ligadas ao gerenciamento dos dados. Essa camada envolve processos complexos, como fusão, análise, processamento e armazenamento de dados.

- Camada de Aplicação: essa camada funciona como um intermediário entre os cidadãos e a camada de gerenciamento de dados. Seu desempenho impacta diretamente a percepção dos usuários e sua satisfação com a Cidades Inteligentes. Ela abrange várias aplicações usadas por cidadãos, servidores públicos e administradores, conectando-se a diferentes departamentos da Cidades Inteligentes. A camada de aplicação inclui todos os sistemas inteligentes que podem ser alimentados com dados. As Aplicações para Cidades Inteligentes são encarregadas de executar as decisões que provêm da camada de gerenciamento de dados.

2.4 Redes de Sensores, IoT e suas Aplicabilidades para Aplicações para Cidades Inteligentes

As redes de sensores sem fio (Wireless Sensor Networks – WSN) têm características e restrições distintas dos sistemas de comunicação convencionais, o que resultou no desenvolvimento de sistemas operacionais específicos para elas (IBRAHIM; MAHDI; YAS, 2021), como, por exemplo, TinyOS (GAY; LEVIS; CULLER, 2007) e Contiki-NG (OIKONOMOU et al., 2022). Tais redes funcionam por meio de tecnologias como Wi-Fi, ZigBee, Bluetooth, 5G, etc.

WSNs são conhecidas como *multi-hop*, uma vez que os dados são transferidos de um nó para outro até alcançarem o destino final. São compostas por sensores e atuadores que

possuem capacidades de auto-organização, porém com capacidade de operar com uma quantidade limitada de energia. WSNs possuem nós que obtém dados automaticamente do ambiente monitorado, o que aprimora a performance dos sistemas de monitoramento que as utilizam. Uma rede de sensores consiste basicamente em uma série de pequenos nós (dispositivos) que se comunicam para identificar eventos ou circunstâncias em um local específico. Cada nó é composto por elementos como processador, memória, transceptor de rádio, fonte de energia e um GPS (IBRAHIM; MAHDI; YAS, 2021).

WSNs são empregadas em várias áreas para acompanhar condições ambientais e de infraestruturas físicas em tempo real. A disposição dos nós depende da configuração necessária para a aplicação. Elas são compostas por vários sensores que trabalham juntos para alcançar um objetivo específico, tomando decisões com base em dados coletados. Esses sensores enviam as informações que capturam para nós centrais chamados estações base. Essas estações reúnem e processam os dados recebidos para possibilitar análises e ações. Esse funcionamento é essencial porque, em muitas aplicações, é necessário o uso de centenas – ou até milhares – de sensores para garantir uma cobertura eficiente e obter resultados mais precisos. As WSNs são econômicas e eficazes, mas apresentam restrições como energia limitada, alcance de comunicação limitado, largura de banda reduzida e capacidade de processamento limitada. Ressalta-se que a administração da rede pode se tornar dispendiosa devido à complexidade do controle de dados e da distribuição (IBRAHIM; MAHDI; YAS, 2021).

As WSNs possuem diversas aplicações como a detecção e rastreamento de veículos, o gerenciamento de desastres e robótica. Elas também são utilizadas no sensoriamento distribuído para fins militares, na detecção de terremotos e no controle de dispositivos como aqueles dispostos em casas inteligentes. Além disso, são úteis na área da saúde, auxiliando em cuidados médicos e no acompanhamento de pacientes (SOHRABY; MINOLI; ZNATI, 2007).

As WSNs também são utilizadas em Aplicações para Cidades Inteligentes. Por exemplo, Ali, Soe e Weller (2015) utilizaram WSNs para realizar o monitoramento da qualidade do ar de escolas em Cidades Inteligentes. As WSNs também são utilizadas para a detecção em tempo real da ocupação de estacionamentos em Cidades Inteligentes conforme o trabalho apresentado por Baroffio et al. (2015). Idwan, Zubairi e Mahmood (2016) propuseram o uso de WSNs para otimizar a gestão de resíduos sólidos em Cidades Inteligentes. A ideia central é utilizar lixeiras inteligentes equipadas com sensores de nível de resíduos, permitindo uma coleta de lixo mais eficiente e econômica. Jain e Shah (2016) apresentaram o uso de WSNs combinadas com algoritmos de aprendizado de máquina para monitorar a poluição em tempo real, detectar anomalias e contribuir para a melhoria da qualidade do ar. Observa-se que tais redes podem ser utilizadas para o monitoramento do ambiente em tempo real em Aplicações para Cidades Inteligentes.

A IoT permite que objetos estejam cada vez mais interconectados. Em 2011, já havia

mais objetos conectados do que pessoas. A IoT afeta vários aspectos da vida urbana, como saúde, segurança e transporte, pois dispositivos de IoT, tais como sensores, estão espalhados pelas cidades. Por exemplo, sensores são utilizados para monitoramento de ciclistas, veículos, estacionamentos públicos e poluição do ar e sonora (ARASTEH et al., 2016).

A IoT é uma inovação tecnológica que permite a comunicação entre objetos do dia a dia, os quais são equipados com microcontroladores, transceptores de comunicação digital e protocolos específicos. Essa conectividade possibilita a troca de informações tanto entre os dispositivos quanto com os usuários, tornando os sistemas mais inteligentes e automatizados. A ideia de IoT tem como objetivo tornar a Internet mais envolvente e onipresente, permitindo acesso e interação com uma vasta gama de aparelhos, tais como sensores, atuadores, telas e veículos. Isso viabiliza o desenvolvimento de diversas aplicações que aproveitam a grande quantidade e variedade de dados gerados por esses dispositivos, possibilitando a criação de novos serviços para cidadãos, empresas e governos. O uso do conceito de IoT em contextos urbanos é particularmente significativo, já que diversos governos promovem a implementação de soluções de TICs para aprimorar a gestão pública. O propósito principal de Cidades Inteligentes é maximizar a utilização de recursos públicos, aprimorar a qualidade dos serviços prestados à população e diminuir gastos operacionais de administrações públicas. Isso pode ser realizado através do estabelecimento de uma infraestrutura de IoT urbana, proporcionando um acesso fácil, unificado e econômico a vários serviços públicos (ZANELLA et al., 2014).

A implementação da IoT atribui à tecnologia o papel de aprimorar a infraestrutura urbana, tornando as áreas urbanas mais eficazes, econômicas e mais agradáveis para se viver. Alguns domínios nos quais a IoT desempenha essencial papel são: casas inteligentes, IoT na agricultura, IoT na hotelaria e turismo e smart grids (KHANG et al., 2023).

Na literatura científica são discutidas aplicações baseadas em IoT para Aplicações para Cidades Inteligentes. Alguns exemplos são mencionados a seguir. No trabalho de Rubí e Gondim (2021) é apresentada uma plataforma baseada em IoT para Cidades Inteligentes que visa garantir interoperabilidade desde a captura de dados até a extração e visualização de conhecimento. A abordagem utiliza tecnologias de Web semântica e a definição de uma ontologia para indicadores ambientais, permitindo a padronização dos dados e facilitando a integração entre diferentes sistemas. A arquitetura da plataforma inclui dispositivos IoT, gateways, características de computação em nuvem e em névoa, otimizando o processamento e a aplicação de técnicas de Big Data. O trabalho menciona desafios de interoperabilidade gerados pelo crescimento da IoT no contexto de Cidades Inteligentes, especialmente no que se refere à coleta, processamento e análise de grandes volumes de dados ambientais.

Shahrour e Xie (2021) discutiram o papel da IoT e do *crowdsourcing* na construção de Cidades Inteligentes. Os resultados do estudo evidenciam que a IoT e o *crowdsourcing*

desempenham um papel fundamental em duas camadas essenciais de Aplicações para Cidades Inteligentes: a camada de coleta de dados e a camada de serviços. Os autores mencionaram que essas camadas garantem a conexão entre o mundo físico e o digital e são consideradas pilares centrais dos projetos de Cidades Inteligentes, destacando então a importância da IoT.

O estudo de Said, Kamal e Afifi (2021) propos o uso de IoT para resolver problemas de reserva de estacionamento em grandes cidades. A ideia principal é criar um sistema inteligente de estacionamento, onde sensores e dispositivos conectados possam ajudar os motoristas a encontrar vagas disponíveis de maneira rápida e eficiente.

Em conclusão, as WSNs e a IoT possibilitam uma ampla gama de aplicações para Cidades Inteligentes. Como outro exemplo, pode-se citar o monitoramento da qualidade da água; nesse caso, sensores espalhados pela cidade coletam dados a qualidade da água, auxiliando na formulação de políticas públicas ambientais e na proteção da saúde da população. Além disso, a IoT também pode ser usada para monitorar e otimizar o transporte público; nesse caso sensores em ônibus ou trens podem fornecer dados sobre lotação e tempo de viagem.

Neste trabalho, em um primeiro momento foi realizada uma RSL a fim de investigar o estado da arte de tolerância a defeitos em SAs e SSCs. Em seguida, foi realizada uma RSL a fim de investigar técnicas de tolerância a defeitos que têm sido utilizadas em Aplicações para Cidades Inteligentes. A primeira revisão auxiliou no entendimento de técnicas e tipos de defeitos também identificados na segunda revisão. A seguir são apresentados os conceitos de dependabilidade e tolerância a defeitos com objetivo de estabelecer os principais termos relacionados as técnicas investigadas neste trabalho.

2.5 Dependabilidade

Antes de abordar o conceito de tolerância a defeitos, é necessário mencionar o conceito de Dependabilidade. Dependabilidade (do inglês, *Dependability*) é a propriedade que define a capacidade de sistemas computacionais entregarem um serviço justificadamente confiável (LAPRIE; AVIZIENIS; KOPETZ, 1992). Para uma descrição sistemática a respeito deste conceito, podem-se mencionar três aspectos: as ameaças, os atributos e os meios pelos quais se pode alcançar dependabilidade (AVIZIENIS et al., 2004). As ameaças à dependabilidade compreendem os conceitos de defeito (do inglês, *fault*), erro (do inglês, *error*) e falha (do inglês, *failure*). Um defeito é definido como a causa física ou algorítmica do erro. Define-se que um sistema está em estado errôneo, ou em erro, quando o sistema apresenta um estado inconsistente, podendo levar a uma falha. Finalmente, define-se falha como o desvio da especificação. Ressalta-se que os conceitos *defeito*, *erro* e *falha* utilizados neste documento estão de acordo com as definições apresentadas no glossário de termos de Engenharia de Software definido pelo IEEE (Institute of Electrical and Electronics

Engineers) (IEEE, 1990), assim como no SWEBOK versão 3.0 (BOURQUE; FAIRLEY, 2014) e também na taxonomia proposta por Avizienis et al. (2004).

O conceito de Dependabilidade engloba os seguintes atributos (AVIZIENIS et al., 2004):

- Disponibilidade (*availability*): o sistema deve ser capaz de prover uma continuidade no oferecimento do serviço;
- Confiabilidade (*reliability*): o sistema deve ser capaz de atender a especificação sob certas condições durante certo período de tempo;
- Segurança (*safety*): o sistema deve ser capaz de de prevenir consequências catastróficas aos usuários ou ao ambiente de operação do sistema;
- Integridade (*integrity*): o sistema deve ser capaz de preservar a consistência de dados;
- Manutenibilidade (*maintainability*): o sistema deve ser estar apto a sofrer modificações e reparos;
- Segurança (*security*): o sistema deve ser capaz de se proteger contra a divulgação de informações para pessoas indevidas.

Com o passar dos anos, diversos meios para alcançar os atributos de dependabilidade têm sido desenvolvidos. Segundo Avizienis et al. (2004), o desenvolvimento de um sistema computacional confiável pode ser alcançado por meio da utilização do seguinte conjunto de quatro técnicas: (a) Técnicas de prevenção de defeitos, que visam a prevenir a ocorrência ou a introdução de defeitos (que, quando executados, podem resultar em erros e subsequentes falhas no sistema). A prevenção de defeitos pode ser alcançada, por exemplo, por meio da aplicação de técnicas de controle de qualidade empregadas durante a etapa de projeto do sistema; (b) Técnicas de tolerância a defeitos, que visam a entregar o serviço corretamente mesmo na presença de defeitos (os quais podem desencadear erros e falhas no sistema); (c) Técnicas de remoção de defeitos, que envolvem, por exemplo, a aplicação de técnicas de Verificação e Validação que podem ser aplicadas durante a fase de projeto e também durante todo ciclo de vida do software; e (d) Técnicas de previsão de falhas, que estimam a ocorrência e a consequência de falhas.

Quase nunca é possível criar sistemas livres de defeitos e decorrentes falhas, especialmente em sistemas com características tais como aquelas presentes em SAs, SSCs e Aplicações para Cidades Inteligentes; tem-se, dessa forma, um fator que pode impactar os usuários deses sistemas. O teste de software possui a finalidade de revelar defeitos. Entretanto, é impossível provar a inexistência de defeitos, pois para isso seria necessário executar o teste de forma exaustiva, ou seja, com todas as entradas possíveis (DELAMARO;

MALDONADO; JINO, 2007). Técnicas de teste são extremamente úteis, mas ainda não conseguem garantir que todos os defeitos sejam eliminados de um sistema (GORLA et al., 2010). Assim, é importante a aplicação de técnicas de tolerância a defeitos a fim de recuperar um sistema mesmo na presença e possível ativação de defeitos.

No caso de Aplicações para Cidades Inteligentes, além dos desafios relacionados à existência de muitas condições de ambiente nas quais tais aplicações operam (multiplicidade de contextos), tais sistemas são geralmente distribuídos, complexos e heterogêneos. Deste modo, todas as possíveis configurações e condições nas quais tais aplicações irão operar podem nunca ser testadas durante a etapa de desenvolvimento, o que pode aumentar o risco de ocorrência de falhas. Aplicações do domínio de Cidades Inteligentes não lidam com um ambiente controlado; dispositivos físicos necessários para o funcionamento de uma aplicação podem estar dispostos no meio ambiente, e portanto, expostos à degradação de material ou acidentes que podem levar a falhas em sensores, dispositivos de comunicação, atuadores, dispositivos de exibição de dados, etc. Uma grande quantidade de componentes podem interagir uns com os outros e erros podem se propagar, gerando falhas em todo o sistema inteligente. Sendo assim, arquiteturas de software e infraestruturas desenvolvidas para utilização em Aplicações para Cidades Inteligentes devem apresentar algum grau de tolerância a defeitos para mitigar a ocorrência de falhas em tais aplicações (NASCIMENTO; OLIVEIRA, 2021).

2.5.1 Tolerância a Defeitos

Um sistema pode conter defeitos que, quando executados, geram estados inconsistentes (ou seja, erros) que podem ou não se tornar perceptíveis (isto é, podem se tornar falhas). Assim sendo, a tolerância que o sistema deve implementar se refere aos defeitos, de forma a continuar operando de forma adequada mesmo que os defeitos sejam executados e erros sejam gerados.[21] Tolerância a Defeitos é um tópico já bastante estudado e desde 1950; técnicas de tolerância a defeitos têm sido adotadas em sistemas computacionais para prover alta confiabilidade em operações de hardware. A tolerância a defeitos destina-se a preservar a entrega de serviço correto mesmo na presença de defeitos ativos (AVIZIENIS, 1967). Lee e Anderson (1990) descrevem quatro fases de aplicação das técnicas de tolerância a defeitos. As primeiras três fases estão relacionadas com a descoberta e o tratamento de erros em um sistema. A quarta fase preocupa-se em lidar com os defeitos que dão origem a esses erros. As quatro fases são:

[21] Observa-se que há divergências históricas na tradução dos conceitos de tolerância a defeitos. Em muitos casos, o termo "*fault tolerance*" é traduzido para "tolerância a falhas", que quando interpretado literalmente, passa uma ideia incorreta, pois a tolerância que o sistema deve apresentar é aos defeitos nele presentes. Nota-se, nesse contexto, que defeitos no sistema podem surgir após a implantação do sistema, como é o caso de um componente de hardware ou software que passa a funcionar fora de suas especificações, ou simplesmente deixa de funcionar completamente.

Fase 1) Detecção de Erros: Falhas não podem ser diretamente detectadas, contudo a manifestação de um defeito pode gerar erros em alguma parte do sistema. Deste modo, o ponto de partida para técnicas de tolerância a defeitos é a detecção de estados errôneos (isto é, inconsistentes) no sistema.

Fase 2) Confinamento e Avaliação: quando um erro é detectado, mais de um estado do sistema pode estar errôneo. Devido ao possível atraso entre a manifestação de um erro e a detecção de suas consequências (falha), dados inválidos podem se propagar dentro do sistema, levando assim a outros erros que podem não ser detectados imediatamente. Deste modo, antes de qualquer ação, é necessário avaliar os limites de propagação de danos causados por erros dentro do sistema.

Fase 3) Recuperação de Erros: após detecção e confinamento, técnicas de recuperação de erros devem ser utilizadas. A recuperação de erros envolve a troca de um estado atual incorreto para um estado livre de erros no qual o sistema pode continuar operando normalmente. Recuperação de erros é um dos aspectos mais importantes de tolerância a defeitos e pode ser realizada de duas formas: por retorno (*backward error recovery*) ou por avanço (*forward error recovery*). Recuperação por retorno envolve a condução do estado errôneo do sistema para um estado anterior consistente. Recuperação por avanço envolve a condução do estado errôneo do sistema para um novo estado que ainda não foi alcançado desde a última manifestação do erro.

Fase 4) Tratamento de Defeitos: se essas três primeiras técnicas conseguirem colocar o sistema em um estado livre de erros, então ele poderá retornar à operação normal, já que o perigo imediato de falha foi evitado. As medidas e mecanismos empregados nas três primeiras fases de tolerância a defeitos estão relacionados a erros no sistema, porém os erros são apenas os sintomas produzidos por um defeito. Abordagens que lidam apenas com erros deixam o defeito que os produziu sem tratamento, podendo levar a outros erros. As técnicas dessa fase visam prover o tratamento para os defeitos eliminando a causa do erro.

Essas são as principais fases que devem ser utilizadas no projeto e implementação de sistemas tolerantes a defeitos, pois estabelecem os princípios básicos para que tolerância a defeitos seja alcançada. Pode haver sobreposição entre as fases, o que pode tornar difícil a identificação de cada etapa separadamente. Por exemplo, para realizar o confinamento e avaliação dos danos causados pelo erro, é necessário realizar primeiramente a etapa de detecção de erros. A ordem em que as fases são realizadas pode variar de sistema para sistema. A detecção de erros normalmente é o ponto de partida para a tolerância a defeitos; já as outras três fases podem estar em qualquer ordem, embora seja comum que a recuperação de erros seja realizada após a etapa de confinamento e avaliação. Deste modo, um dos passos no projeto de um sistema tolerante a defeitos é decidir como essas fases serão aplicadas e implementadas no sistema. O projeto de um sistema tolerante a defeitos é um processo interativo que envolve identificar possíveis erros e avaliar métodos

alternativos de implementar tolerância a defeitos (LEE; ANDERSON, 1990).

2.6 Considerações Finais

Nesta seção foram apresentadas algumas definições para Cidades Inteligentes, fornecendo vários exemplos de Aplicações para Cidades Inteligentes que têm sido desenvolvidas e utilizadas em diferentes cidades. Delineou-se também a relação entre SAs, SSCs e Aplicações para Cidades Inteligentes. Além disso, foram apresentados os conceitos de dependabilidade e tolerância a defeitos. Neste trabalho foram realizadas duas revisões sistemáticas da literatura (RSLs) a fim de elencar técnicas de tolerância a defeitos que têm sido propostas ou aplicadas em Aplicações para Cidades Inteligentes, ou em sistemas que possuem características similares a elas. No próximo capítulo é apresentada em detalhes a RSL sobre técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes. Apresenta-se também um resumo de uma RSL sobre tolerância a defeitos em Sistemas Adaptativos e Sensíveis ao Contexto, bem como a relação desses sistemas com Aplicações para Cidades Inteligentes.

Capítulo 3

Estado da Arte sobre Tolerância a Defeitos para Aplicações para Cidades Inteligentes

3.1 Considerações Iniciais

Neste trabalho foi realizado um estudo secundário, mais especificamente uma Revisão Sistemática da Literatura (RSL), com o objetivo de caracterizar o estado da arte e identificar direções de pesquisa com relação a técnicas de tolerância a defeitos propostas ou utilizadas em Aplicações para Cidades Inteligentes. Neste capítulo, primeiramente, na Seção 3.2, apresentam-se trabalhos relacionados a esta RSL, os quais consistem em estudos secundários (sistemáticos e não-sistemáticos) que abordam técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes. Em seguida, na Seção 3.3, apresenta-se o protocolo e a metodologia utilizados na condução da RSL, e nas Seções 3.4 até 3.7 apresentam-se os resultados obtidos de acordo com o esquema de classificação adotado. Respostas às questões de pesquisa elencadas são apresentadas na Seção 3.8, seguidas por discussões adicionais (Seção 3.9) e direções de pesquisa identificadas (Seção 3.10). A RSL foi publicada em formato de um artigo em periódico (de Souza et al., 2025).

3.2 Trabalhos Relacionados

Nesta seção são apresentados os estudos secundários sistemáticos e não sistemáticos que mencionam técnicas de tolerância a defeitos propostas para ou empregadas em Apli-

cações para Cidades Inteligentes. Tais estudos foram identificados durante a seleção dos estudos primários recuperados na RSL.

No trabalho de Gharaibeh et al. (2017), que seguiu um abordagem não-sistemática para recuperação, seleção e classificação de estudos, são descritas técnicas para garantir segurança e privacidade de dados em Aplicações para Cidades Inteligentes. Os autores mencionaram diferenças entre intermediários de mensagens (*message brokers*), tais como Kafka, RabbitMQ e Mosquitto, que são tecnologias que como consequência de sua utilização fornecem capacidades de tolerância a defeitos. Os autores, no entanto, não empregaram uma abordagem sistemática para coletar e analisar os estudos. Além disso, Gharaibeh et al. (2017) não apresentam estratégias que não estavam incluídas nas ferramentas específicas que discutiram.

Du et al. (2019) apresentaram um estudo não sistemático com o objetivo de identificar e discutir estratégias para a identificação de falhas e anomalias durante a fase de sensoriamento de Aplicações para Cidades Inteligentes. Putra, Putra e Kurniawan (2018) descreveram ferramentas de análise de Big Data, como Spark e Hadoop, que têm sido usadas no domínio das Aplicações para Cidades Inteligentes para o tratamento de falhas em nós, corrupção de dados e problemas de rede. Como consequência da utilização de tais ferramentas, capacidades de tolerância a defeitos são adicionadas. Ainda no contexto de Big Data, o estudo de Putra, Putra e Kurniawan (2018) é uma revisão sistemática –portanto um estudo sistemático – e a string de busca foi restrita aos seguintes termos: *"big data analytics"* e *"smart city"*. Essa string de busca é capaz de recuperar estudos que abordam Aplicações para Cidades Inteligentes, porém apenas no escopo de Big Data. Ou seja, embora mencionem tratamento de falhas, os autores lidaram com uma quantidade de estudos delimitada pelo escopo mencionado.

Javadzadeh e Rahmani (2020) realizaram uma revisão sistemática da literatura sobre Aplicações para Cidades Inteligentes baseadas em computação em neblina (do inglês, *fog computing*). Os autores discutiram duas plataformas que implementam tolerância a defeitos. O estudo seguiu um processo rigoroso para a seleção e análise dos estudos recuperados; no entanto, teve seu foco apenas na tecnologia de computação em neblina, o que se reflete na string de busca usada, a saber: *("edge" ou "fog") e (("smart city") ou ("smart cities"))*. Deste modo, o conjunto de estudos analisado também se tornou limitado pela string de busca. Syed et al. (2021) revisaram extensivamente, por meio de um *survey*, vários aspectos da infraestrutura, de diferentes tipos de aplicações e tecnologias de rede e computação empregadas em Aplicações para Cidades Inteligentes. Os autores mencionaram estratégias para monitorar e detectar falhas nesse tipo de aplicação. No entanto, o estudo é um estudo não-sistemático que não revisa a aplicação de tolerância a defeitos, assim como não menciona estudos que abordaram recuperação de erros e tratamento de defeitos.

A principal diferença entre o estudo conduzido nesta RSL e os estudos secundários

relacionados são os objetivos da revisão e a string de busca que foi projetada neste trabalho para cumprir os objetivos, os quais se resumem em identificar técnicas de tolerância a defeitos, e tipos de defeitos, erros e falhas em Aplicações para Cidades Inteligentes.

3.3 Protocolo da RSL

Nesta seção apresentam-se detalhes do protocolo de pesquisa projetado e executado na RSL, incluindo seus objetivos, questões de pesquisa, strings de busca, etapas de busca por estudos, seleção, extração de dados e classificação. Este protocolo foi criado conforme práticas bem estabelecidas para condução de revisões sistemáticas (KITCHENHAM, 2004).

3.3.1 Objetivo e Questões de Pesquisa

O principal objetivo desta RSL é caracterizar o estado atual da pesquisa sobre aplicação de técnicas de Tolerância a Defeitos para Aplicações para Cidades Inteligentes e, assim, abrir caminho para a pesquisa e desenvolvimento de aplicações robustas para este domínio. Para isso, desenvolvemos as seguintes Questões de Pesquisa (QP):

- **QP1** *Quais técnicas de tolerância a defeitos têm sido empregadas em Aplicações para Cidades Inteligentes?*
- **QP2** *Quais são os tipos de defeitos, erros e falhas envolvidos em Aplicações para Cidades Inteligentes, e quais deles foram tratados por técnicas de tolerância a defeitos?*
- **QP3** *Quão madura é a pesquisa sobre tolerância a defeitos para Aplicações para Cidades Inteligentes?*

A QP2 complementa a QP1 ao fornecer ao leitor informações úteis para conceber ou aplicar técnicas de tolerância a defeitos. Da mesma forma, a QP3 complementa a QP1 ao fornecer ao leitor uma visão da aplicabilidade prática de técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes.

3.3.2 Strings de Pesquisa e Motor de Busca

Inicialmente foram realizadas buscas automáticas. Em seguida, realizaram-se buscas manuais para recuperar estudos relevantes. Uma abordagem híbrida foi seguida conforme proposto por Mourao et al. (2017). Nessa abordagem, um motor de busca é escolhido pelo pesquisador, sendo esse um motor conhecido por sua capacidade de recuperar uma ampla gama de estudos, e, em seguida, buscas manuais são aplicadas utilizando-se os estudos recuperados e selecionados via busca automática. O motor de busca *Scopus*[1] foi selecionado por abranger repositórios tradicionais de Ciência da Computação. Foram definidos

[1] <https://www.scopus.com/home.uri> - acessado em novembro de 2024.

dois conjuntos de palavras-chave para formar strings de busca automática, denominadas String 1 e String 2. A String 1 foi criada com o objetivo de recuperar estudos relacionados à (QP1), enquanto a String 2 foi criada com o objetivo de recuperar estudos que caracterizam defeitos, erros e falhas (QP2) no contexto de Aplicações para Cidades Inteligentes. Estudos recuperados com ambas as strings também foram empregados na análise referente à QP3. A língua escolhida para as strings foi a língua inglesa, considerando-se que os principais veículos de publicação possuem a maioria dos estudos publicados em inglês. As strings foram ajustadas para buscarem estudos por título, resumo e palavras-chave, e ambas são apresentadas a seguir:

String 1: *(("smart city" OR "smart cities" OR "smart solution" OR "smart solutions") AND ("fault-tolerance" OR "fault tolerance" OR "fault-tolerant" OR "fault tolerant" OR "dependability" OR "dependable" OR "error recovery" OR "error handling" OR "reliable" OR "resilient"))*

String 2: *(("smart city" OR "smart cities" OR "smart solution" OR "smart solutions") AND ("fault type" OR "error type" OR "failure type" OR "fault characterisation" OR "fault characterization" OR "fault classification" OR "error characterisation" OR "error characterization" OR "error classification" OR "failure characterisation" OR "failure characterization" OR "failure classification"))*

As buscas manuais consistiram em aplicar a técnica de *snowballing* (WOHLIN, 2014), tanto para frente (*forward*) quanto para trás (*bakward*). O objetivo foi cobrir uma amostra mais ampla da literatura. O *snowballing backward* consistiu em analisar cada referência utilizada nos estudos selecionados via busca automática. Já durante o *snowballing foward*, foram analisados os estudos que citaram os estudos selecionados via busca automática. Para isso, utilizou-se o motor de busca Google Scholar.[2] Ambas as estratégias de busca via *snowballing* foram executadas em um único nível de profundidade (ou seja, uma iteração) e todas as etapas de seleção de estudos foram efetuadas nos estudos provenientes das buscas com essa técnica.

3.3.3 Etapas para a Seleção dos Estudos

Um estudo foi selecionado se atendeu a pelo menos um dos critérios de inclusão I1 e I2, e atendeu ao critério I3 (listados a seguir). Ademais, um estudo foi excluído se atendeu a pelo menos um critério de exclusão E1 ou E2 (também listados a seguir). Estudos secundários ou terciários foram analisados e, quando aplicável, classificados como trabalhos relacionados e descritos na Seção 3.2.

- ❑ I1: O estudo define ou aplica técnicas de tolerância a defeitos para Aplicações para Cidades Inteligentes.

[2] <https://scholar.google.com/> - acessado em novembro de 2024.

- ❏ I2: O estudo define catálogos, classificações ou taxonomias de defeitos, erros ou falhas para Aplicações para Cidades Inteligentes.
- ❏ I3: O estudo é um estudo primário.
- ❏ E1: O estudo não está escrito em inglês.
- ❏ E2: O estudo não foi revisado por pares.

A ferramenta StArt –State of the Art through Systematic Review (HERNANDES et al., 2012)[3] foi adotada para apoiar as etapas de seleção inicial e final dos estudos. Durante a seleção inicial a ferramenta foi utilizada para unir os arquivos em formato Bibtex provenientes do motor de busca, o que permitiu a leitura dos títulos e resumos de todos os estudos em sequência. Dessa forma, os critérios de inclusão e exclusão foram aplicados inicialmente na primeira fase e, posteriormente, na segunda fase, durante a leitura do texto completo. A seleção dos estudos foi realizada em quatro etapas, conforme descritas a seguir:

- ❏ **Filtragem de estudos duplicados (Etapa 1)**: Remoção de estudos duplicados recuperados pelo motor de busca.
- ❏ **Seleção inicial dos estudos (Etapa 2)**: Aplicação dos critérios de inclusão e exclusão no título, resumo e palavras-chave de cada estudo.
- ❏ **Seleção final (Etapa 3)**: Análise do conteúdo completo dos estudos e nova aplicação dos critérios de inclusão e exclusão. Indecisões quanto à seleção dos estudos foram resolvidas entre dois ou mais pesquisadores envolvidos.
- ❏ **Identificação de sobreposições (Etapa 4)**: As sobreposições foram identificadas, por exemplo, estudos que foram atualizados ou ampliados por seus autores, ou um novo estudo que substitui um estudo anterior. Quando sobreposições foram identificadas, utilizou-se a versão mais recente publicada.

3.3.4 Formulário para a Extração de Dados

Após a etapa de seleção final os dados dos estudos foram extraídos e em seguida uma análise sobre tais dados foi feita a fim de classificar os estudos de acordo com o esquema de classificação proposto. A seguir são apresentadas as informações extraídas dos estudos selecionados:

- ❏ Um resumo do estudo, incluindo seu contexto e objetivo, metodologia e conclusão.
- ❏ Ano de publicação.

[3] https://www.lapes.ufscar.br/resources/tools-1/start-1 - acessado em novembro de 2024.

- ❏ Tipo de pesquisa (WIERINGA et al., 2006).
- ❏ Tipo de contribuição (ISO/IEC/IEEE, 2017).
- ❏ Domínios e subdomínios de aplicação (SÃNCHEZ-CORCUERA et al., 2019).
- ❏ Tipo de arquitetura.
- ❏ Técnica de tolerância a defeitos.
- ❏ Descrição de defeitos (buscando-se pelo termo *fault*), erros (buscando-se pelo termo *error*) e falhas (buscando-se pelo termo *failure*), caso tais informações estivessem disponíveis no estudo.

3.3.5 Processo de Classificação dos Estudos

Primeiramente, em uma *classificação inicial*, os estudos foram classificados e quantificados por ano de publicação, local de publicação (periódico, congresso ou workshop), tipo de pesquisa, tipo de contribuição, domínios e subdomínios de aplicação, e tipo de arquitetura. Em uma *classificação avançada*, os estudos foram agrupados e classificados pelo tipo de técnica de tolerância a defeitos, e por tipos de defeitos, erros e falhas. A classificação inicial foi realizada da seguinte maneira:

- ❏ **Tipo de Pesquisa**: Foi seguida a classificação desenvolvida por Wieringa et al. (2006). A classificação inclui seis categorias de tipo de pesquisa, a saber: pesquisa de validação, pesquisa de avaliação, proposta de solução, artigo filosófico, artigo de opinião e artigo de experiência. A classificação efetuada nesta RSL foi apoiada por uma tabela de decisão proposta por Petersen, Vakkalanka e Kuzniarz (2015).
- ❏ **Tipo de Contribuição**: Para esta classificação, foram seguidas as classes de contribuição conforme propostas por Petersen, Vakkalanka e Kuzniarz (2015), sendo elas: um processo, um método, um modelo, uma ferramenta ou uma métrica. Nesta RSL, a classificação do tipo de contribuição foi realizada utilizando-se o documento *ISO/IEC/IEEE International Standard - Systems and Software Engineering – Vocabulary* (ISO/IEC/IEEE, 2017).
- ❏ **Domínios e Subdomínios de Aplicação**: Tal classificação fornece o tipo de aplicação abordada no estudo em termos do espaço da cidade que a funcionalidade da aplicação atendia. Para esta classificação foi utilizada a classificação estabelecida por Sãnchez-Corcuera et al. (2019), que inclui os seguintes domínios: *Relacionados ao Governo*, *Relacionados ao Cidadão*, *Relacionados aos Negócios*, e *Relacionados aos Negócios*. Cada domínio inclui um conjunto de subdomínios que estão listados no protocolo da RSL que está disponível online (SOUZA, 2024).

- **Tipo de Arquitetura**: Essa classificação fornece o tipo de arquitetura de sistema para o qual a solução (isto é, a contribuição) foi proposta em relação às tecnologias: Edge Computing, Cloud Computing, e Fog Computing. O tipo de arquitetura foi coletado apenas se mencionado explicitamente pelos autores (ou seja, não se inferiu a respeito do tipo de arquitetura, nem mesmo com base em esquemas que representam as aplicações no estudo).

  Após a classificação inicial, foi realizada uma classificação avançada, utilizada para responder às questões de pesquisa, e conduzida da seguinte maneira:

- **Técnicas de tolerância a defeitos**: Essa classificação diz respeito às características da técnica de investigada no contexto de Aplicações para Cidades Inteligentes. É um elemento central para a resposta à QP1. Nesta RSL, esta classificação se baseia exclusivamente no conjunto de estudos selecionados. Dessa forma, analisando-se os estudos, identificaram-se três categorias de técnicas bem conhecidas, a partir das quais foi elaborada uma lista de subcategorias. As categorias encontradas são: Diversidade (D), Reconfiguração de Sistema (RS) e Retry (R) (LEE; ANDERSON, 1990).

- **Tipos de Falhas, Erros ou Defeitos**: Essa classificação diz respeito às características dos defeitos, erros e falhas que foram descritos nos estudos selecionados. Defeitos, erros e falhas foram coletados e agrupados em diferentes tipos. Quando o estudo não mencionava defeitos, buscou-se por erros e/ou falhas mencionadas como alvo da solução de tolerância a defeitos proposta. Procurou-se encontrar a causa que levou à necessidade de recuperação do sistema. Essa classificação é um elemento chave para responder a QP2 e, novamente, baseia-se exclusivamente no conjunto de estudos selecionados.

3.4 Resultados das Buscas Automáticas e Manuais

Na Tabela 3 são apresentados o número de estudos recuperados e analisados durante cada busca, realizada em diferentes rodadas em diferentes períodos de tempo, e o número de estudos selecionados após as etapas do processo de seleção. A primeira coluna da tabela indica o número de rodadas e as datas em que cada rodada foi realizada, a segunda apresenta o tipo de busca, e a terceira coluna exibe o número de estudos recuperados. As três últimas colunas apresentam o número de estudos selecionados em cada etapa de seleção (etapas 2, 3 e 4, conforme descrito na Seção 3.3.3). Na coluna 2 da tabela, conjuntamente com o tipo de busca realizado, indica-se o nome da rodada de busca nas quais estudos foram selecionados.

Três rodadas de busca automática foram realizadas nas seguintes datas: 08/Mar/2021 (rotulada na Tabela 3 como Automatica1), 19/Jan/2023 (rotulada na tabela como Automatica2), e 08/Jul/2023 (rotulada na tabela como Automatica3). As buscas manuais empregando *snowballing backward* e *forward* foram realizadas nos 15 estudos obtidos a

partir da busca automática (Automatica1) em 02/Jun/2023 e 23/Jun/2023. O *snowballing backward* aplicado nos 24 estudos selecionados em R5 e R7 foi realizado em 06/Nov/2023. Em seguida, para buscar novos estudos, uma nova rodada de *snowballing forward* foi realizada nos 16 estudos selecionados na Automatica1 em 17/Nov/2023. No total, 43 estudos foram selecionados. Nas Tabelas 4 e 5 são listados os 43 estudos selecionados. A seguir, apresenta-se os detalhes da classificação inicial dos estudos.

Data de Busca	**Busca**	**# de estudos recuperados**	**Etapa 2 (Pre-(Seleção)**	**Estapa 3 (Seleção final)**	**Etapa 4 Sobreposições**
R1 08/Mar/2021	Automatica1 - String 1	1392	232	15	0
R2 08/Mar/2021	Automatica1 - String 2	11	1	0	0
R3 02/Jun/2021	Snowballing backward (15 estudos a partir de Automatica1)	1465	91	0	0
R4 23/Jun/2021	Snowballing forward (15 estudos a partir de Automatica1)	1074	46	3	1
R5 19/Jan/2023	Automatica2 - String 1	740	190	22	1
R6 19/Jan/2023	Automatica2 - String 2	3	1	0	0
R7 08/Jul/2023	Automatica3 - String 1	224	37	2	0
R8 08/Jul/2023	Automatica3 - String 2	0	0	0	0
R9 06/Nov/2023	Snowballing backward (24 estudos a partir de R5 e R7)	906	0	1	0
R10 17/Nov/2023	Snowballing forward (15 estudos a partir de R1)	331	46	0	0

Tabela 3 – Número de estudos recuperados em cada rodada e selecionados após as etapas do processo de seleção.

3.5 Classificação Inicial dos Estudos

Tal classificação concentra-se em apresentar: i) a distribuição de estudos por ano, ii) veículo de publicação, iii) tipo de pesquisa, iv) tipo de contribuição, v) domínios e subdomínios de aplicação e vi) tipo de arquitetura. Logo, os resultados são apresentados nas seções a seguir.

3.5.1 Estudos por Ano e Veículo de Publicação

Na Figura 1 é apresentada a distribuição dos 43 estudos por ano. O primeiro estudo foi publicado em 2013 e o número de estudos publicados tem aumentado desde então. Nota-se que os dados para 2023 são parciais devido às datas em que as pesquisas foram realizadas. Essa distribuição pode sugerir que Cidades Inteligentes têm sido um tópico debatido nos últimos anos como um tópico relevante na comunidade científica.

Todos os estudos selecionados foram revisados por pares, ou seja, especialistas da mesma área de conhecimento analisam criticamente o conteúdo dos estudos. Do conjunto de 43 estudos selecionados, 31 foram publicados em periódicos e 12 em anais de eventos científicos. Na Tabela 6 são listados os veículos de publicação identificados. A coluna “tipo” indica o tipo do veículo de publicação na qual a sigla “j” indica um periódico e a

Estudo	Título	Ano	Motor/Base
(MORA-MORA et al., 2015)	A computational architecture based on RFID sensors for traceability in smart cities	2015	MDPI
(NANDURY; BEGUM, 2015)	Smart WSN-based ubiquitous architecture for smart cities	2015	IEEE **(SB)**
(SASU; PUIU; NECHIFOR, 2016) (extends (PUIU et al., 2016))	Fault recovery mechanism for smart city environments	2016	IEEE **(SF)**
(PUIU et al., 2016)	CityPulse: Large Scale Data Analytics Framework for Smart Cities	2016	IEEE
(HE et al., 2016)	Towards smarter cities: A self-healing resilient Microgrid Social Network	2016	IEEE
(ABREU et al., 2017)	A resilient Internet of Things architecture for smart cities	2017	SpringerLink
(HASEBE et al., 2017)	Traffic management for last-mile public transportation systems using autonomous vehicles	2017	IEEE
(LIU et al., 2017)	A fault-Tolerant Mobile Sensing Information Gathering Center (MSIGC) using public transport buses to instrument a smart city	2017	IEEE
(ALKADY et al., 2019)	Reliable FPGA-based network architecture for smart cities	2019	IEEE
(HAKIRI; GOKHALE; BERTHOU, 2019)	Software-defined Wireless Mesh Networking for Reliable and Real-time Smart City Cyber Physical Applications	2019	ACM Digital Library
(POWAR et al., 2019)	Sensor networks for hydrometric monitoring of urban watercourses	2019	ScienceDirect
(SHAH et al., 2019)	Towards Disaster Resilient Smart Cities: Can Internet of Things and Big Data Analytics Be the Game Changers?	2019	IEEE
(BAWANY; SHAMSI, 2019)	SEAL: SDN based secure and agile framework for protecting smart city applications from DDoS attacks	2019	Science Direct
(MOHAMED; AL-JAROODI; JAWHAR, 2019)	Towards fault tolerant fog computing for IoT-based smart city applications	2019	IEEE
(HAMDAOUI et al., 2020)	IoTShare: A Blockchain-Enabled IoT Resource Sharing On-Demand Protocol for Smart City Situation-Awareness Applications	2020	IEEE
(ALZOMAN; ALENAZI, 2020)	Exploiting SDN to Improve QoS of Smart City Networks Against Link Failures	2020	IEEE **(SF)**
(SHAMSI, 2020)	Resilience in Smart City Applications: Faults, Failures, and Solutions	2020	IEEE **(SF)**
(MODARRESI; SYMONS, 2020)	Resilience and technological diversity in smart homes: A graph-theoretic approach to modeling IoT systems with integrated heterogeneous networks	2020	SpringerLink
(YUAN et al., 2021)	An alternative reliability method to evaluate the regional traffic congestion from GPS data obtained from floating cars	2021	Wiley Online Library
(AHMAD et al., 2021)	Complex problems solution as a service based on predictive optimization and tasks orchestration in smart cities	2021	Tech Science Press
(ALJOHANI; ALENAZI, 2021)	Mpresisdn: Multipath resilient routing scheme for sdn-enabled smart cities networks	2021	MDPI
(BIDI; MOVAHEDI; MOVAHEDI, 2021)	A fog-based fault-tolerant and QoE-aware service composition in smart cities	2021	Wiley
(KHAN et al., 2021)	Privacy preserving data aggregation with fault tolerance in fog-enabled smart grids	2021	Science Direct
(KURESHI et al., 2021)	Use Case of Building an Indoor Air Quality Monitoring System	2021	IEEE
(KIM; BEN-OTHMAN, 2021)	FAMU: Fault-Tolerant Mutual Assisted Virtual Emotion Barrier System Using Intelligent Smart UAVs	2021	IEEE
(TANCEV; TORO, 2021)	Sequential recalibration of wireless sensor networks with (stochastic) gradient descent and mobile references	2021	ScienceDirect
(WANG; LIU; CHOO, 2021)	Fault-Tolerant Multisubset Aggregation Scheme for Smart Grid	2021	IEEE
(MOHAPATRA; RATH, 2021)	A fault tolerant routing scheme for advanced metering infrastructure: an approach towards smart grid	2021	Springer Link
(PASTóRIO; CAMARGO, 2021)	Geolocation Techniques in LoRaWan Networks as a Fault Tolerance Approach in GPS-Based Tracking Devices	2021	IEEE

Tabela 4 – Os 43 estudos selecionados a partir de buscas automáticas e manuais, organizados por ano de publicação - parte 1 (SB: *Snowballing Backward*; SF: *Snowballing Forward*).

sigla "c" indica um evento científico. Os periódicos mais recorrentes foram *Sensors*, *IEEE Access* e *Internet of Things Journal*. A evento mais recorrente foi *International Smart*

Estudo	**Título**	**Ano**	**Motor/Base**
(ALI et al., 2022)	Mathematical Modeling and Validation of Retransmission-Based Mutant MQTT for Improving Quality of Service in Developing Smart Cities	2022	Science Direct
(ANUPONG et al., 2022)	Towards a high precision in AMI-based smart meters and new technologies in the smart grid	2022	Science Direct
(COSTA; NASSAR; DANTAS, 2022)	FOCUSeR: A Fog Online Context-Aware Up-to-Date Sensor Ranking Method	2022	MDPI
(FERRARI et al., 2022)	On the Use of LoRaWAN and Cloud Platforms for Diversification of Mobility-as-a-Service Infrastructure in Smart City Scenarios	2022	IEEE
(HASEEB et al., 2022)	Trust Management With Fault-Tolerant Supervised Routing for Smart Cities Using Internet of Things	2022	IEEE
(YANG; PARK; KIM, 2022)	Collaborative Reliable Event Transport Based on Mobile-Assisted Sensing in Urban Digital Twin	2022	MDPI
(YU et al., 2022b)	Distributed Fractional-Order Intelligent Adaptive Fault-Tolerant Formation-Containment Control of Two-Layer Networked Unmanned Airships for Safe Observation of a Smart City	2022	IEEE
(YU et al., 2022a) (extends (YU et al., 2022b))	Distributed Adaptive Fault-Tolerant Time-Varying Formation Control of Unmanned Airships With Limited Communication Ranges Against Input Saturation for Smart City Observation	2022	IEEE
(JESUS et al., 2022)	A dependability-aware approach for dynamic mobile sink repositioning in smart cities applications	2022	IEEE
(KHAN; KIM; PARK, 2022)	Leveraging Machine Learning for Fault-Tolerant Air Pollutants Monitoring for a Smart City Design	2022	MDPI
(KHARCHENKO et al., 2022)	UAV Fleet as a Dependable Service for Smart Cities: Model-Based Assessment and Application	2022	MDPI
(MANOGARAN; NGUYEN, 2022)	Displacement-Aware Service Endowment Scheme for Improving Intelligent Transportation Systems Data Exchange	2022	IEEE
(MIRZA; GEORGAKOPOULOS; YAVARI, 2023)	Cyber-Physical-Social Awareness Platform for Comprehensive Situation Awareness	2023	MDPI
(ALI; ZAFAR, 2023)	Improved End-to-end service assurance and mathematical modeling of message queuing telemetry transport protocol based massively deployed fully functional devices in smart cities	2023	Science Direct

Tabela 5 – Os 43 estudos selecionados a partir de buscas automáticas e manuais, organizados por ano de publicação - parte 2 (SB: *Snowballing Backward*; SF: *Snowballing Forward*).

Cities Conference. Nota-se que a maioria dos estudos foi publicada em periódicos, o que pode indicar a relevância acadêmica e o impacto científico do tema alvo, e a confiabilidade acadêmica dos estudos selecionados.

3.5.2 Tipo de Pesquisa

Dentre os 43 estudos selecionados, 5 estudos foram classificados como Pesquisa de Avaliação. Um desses estudos foi conduzido por Mora-Mora et al. (2015), que utilizou o campus da Universidade de Alicante na Espanha para implantar e testar sua abordagem. O objetivo do trabalho foi melhorar os serviços oferecidos pelo campus. Outro estudo classificado como Pesquisa de Avaliação foi realizado por Khan et al. (2021), que utilizou dados reais do banco de dados Irish Social Science Data Archive. Os estudos de Sasu, Puiu e Nechifor (2016), Puiu et al. (2016) e Yuan et al. (2021) também utilizaram dados reais na avaliação.

Seis estudos foram classificados como Propostas de Solução, e 32 foram classificados como Pesquisa de Validação. Na Tabela 7 são apresentados os estudos de cada tipo de

Estudos	**Nome do Veículo**	**Tipo**
(MORA-MORA et al., 2015; MIRZA; GEORGAKOPOULOS; YAVARI, 2023)	Sensors	j
(NANDURY; BEGUM, 2015)	International Conference on Advances in Computing, Communications and Informatics (ICACCI)	c
(PUIU et al., 2016; SHAH et al., 2019)	IEEE Access	j
(HE et al., 2016)	IEEE General Meeting Power & Energy Society	c
(SASU; PUIU; NECHIFOR, 2016)	International Conference on Intelligent Engineering Systems (INES)	c
(LIU et al., 2017)	International Conference on Advanced Infocomm Technology (ICAIT)	c
(ABREU et al., 2017)	Annals of Telecommunications	j
(HASEBE et al., 2017; JESUS et al., 2022)	International Smart Cities Conference (ISC2)	c
(ALKADY et al., 2019)	International Conference on Microelectronics (ICM)	c
(BAWANY; SHAMSI, 2019)	Journal of Network and Computer Applications	j
(HAKIRI; GOKHALE; BERTHOU, 2019)	International Conference on Real-Time Networks and Systems (RTNS)	c
(POWAR et al., 2019)	International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT and AI (HONET-ICT)	c
(MOHAMED; AL-JAROODI; JAWHAR, 2019)	Annual Computing and Communication Workshop and Conference (CCWC)	c
(HAMDAOUI et al., 2020; HASEEB et al., 2022)	Internet of Things Journal	j
(ALZOMAN; ALENAZI, 2020)	International Conference on Software Defined Systems (SDS)	c
(SHAMSI, 2020)	Journal of Network and Computer Applications	j
(MODARRESI; SYMONS, 2020)	Journal of Ambient Intelligence and Humanized Computing	j
(KHAN et al., 2021)	Electronics	j
(AHMAD et al., 2021)	Computers, Materials & Continua	j
(ALJOHANI; ALENAZI, 2021)	Applied Sciences	j
(BIDI; MOVAHEDI; MOVAHEDI, 2021)	Transactions on Emerging Telecommunications Technologies	j
(KIM; BEN-OTHMAN, 2021)	Transactions on Vehicular Technology	j
(KURESHI et al., 2021)	World Forum on Internet of Things (WF-IoT)	
(MOHAPATRA; RATH, 2021)	Cluster Computing	j
(PASTóRIO; CAMARGO, 2021)	South American Colloquium on Visible Light Communications (SACVLC)	c
(TANCEV; TORO, 2021)	Measurement: Sensors	j
(ALI et al., 2022)	Alexandria Engineering Journal	j
(ANUPONG et al., 2022)	Sustainable Computing: Informatics and Systems	j
(COSTA; NASSAR; DANTAS, 2022)	Journal of Sensor and Actuator Networks	j
(FERRARI et al., 2022)	Transactions on Instrumentation and Measurement	j
(KHAN; KIM; PARK, 2022)	Electronics	j
(KHARCHENKO et al., 2022)	Smart Cities	j
(MANOGARAN; NGUYEN, 2022)	Transactions on Intelligent Transportation Systems	j
(WANG; LIU; CHOO, 2021)	Transactions on Industrial Informatics	j
(YANG; PARK; KIM, 2022)	Electronics	j
(YU et al., 2022b)	Transactions on Cybernetics	j
(YU et al., 2022a)	Transactions on Neural Networks and Learning Systems	j
(YUAN et al., 2021)	IET Smart Cities	j
(ALI; ZAFAR, 2023)	Alexandria Engineering Journal	j

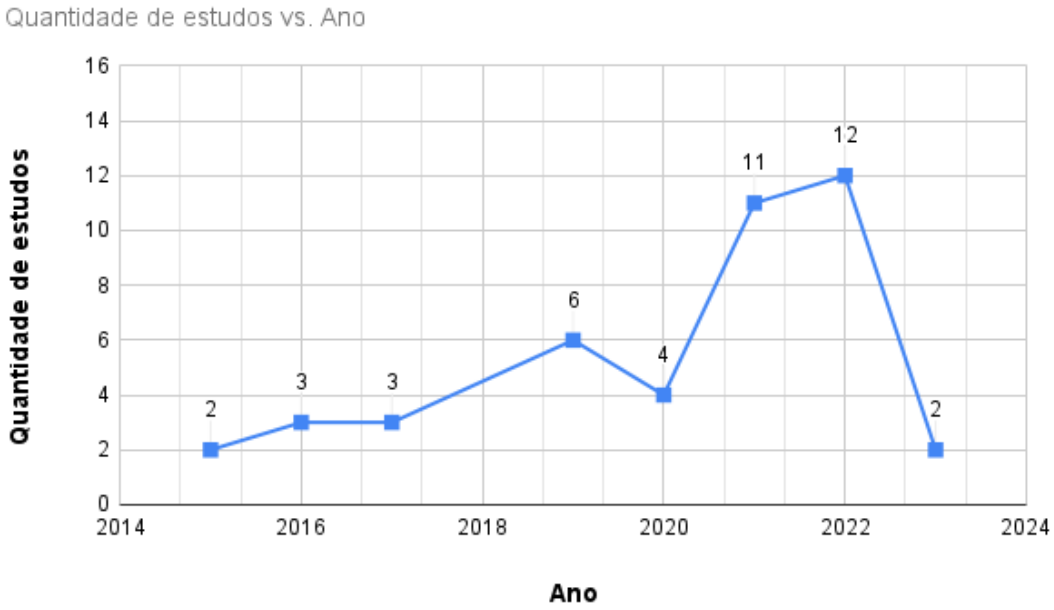
Tabela 6 – Veículos de publicação dos 43 estudos selecionados.

pesquisa. Ressalta-se a quantidade limitada de estudos que documentaram Pesquisas de Avaliação. Tal tipo de pesquisa busca avaliar a solução conforme foi utilizada no mundo real, por exemplo, na indústria ou em cidades (seguindo o contexto deste trabalho), o que fornece ideias sobre como melhorar a solução sob investigação para melhor atender aos objetivos da solução e das partes interessadas (*stakeholders*) na mesma.

3.5.3 Tipo de Contribuição

De acordo com Petersen, Vakkalanka e Kuzniarz (2015), que propuseram um guia para condução de mapeamentos sistemáticos em engenharia de software, é possível realizar uma

Figura 1 – Distribuição de estudos por ano.



Fonte: Produzida pela autora.

Tipo de Pesquisa	**Estudos**
Pesquisa de Avaliação	Mora-Mora et al. (2015), Khan et al. (2021), Puiu et al. (2016), Sasu, Puiu e Nechifor (2016), Yuan et al. (2021)
Proposta de Solução	Nandury e Begum (2015), Abreu et al. (2017), Hasebe et al. (2017), Alkady et al. (2019), Mohamed, Al-Jaroodi e Jawhar (2019), Shamsi (2020)
Pesquisa de Validação	He et al. (2016), Liu et al. (2017), Hakiri, Gokhale e Berthou (2019), Bawany e Shamsi (2019), Powar et al. (2019), Shah et al. (2019), Modarresi e Symons (2020), Hamdaoui et al. (2020), AlZoman e Alenazi (2020), Haseeb et al. (2022), Ahmad et al. (2021), Kim e Ben-Othman (2021), Kureshi et al. (2021), Mohapatra e Rath (2021), Pastório e Camargo (2021), Tancev e Toro (2021), Wang, Liu e Choo (2021), Bidi, Movahedi e Movahedi (2021), Aljohani e Alenazi (2021), Yang, Park e Kim (2022), Yu et al. (2022b), Yu et al. (2022a), Ali et al. (2022), Anupong et al. (2022), Costa, Nassar e Dantas (2022), Ferrari et al. (2022), Jesus et al. (2022), Khan, Kim e Park (2022), Kharchenko et al. (2022), Manogaran e Nguyen (2022), Mirza, Georgakopoulos e Yavari (2023), Ali e Zafar (2023)

Tabela 7 – Classificação a respeito do tipo de pesquisa dos estudos identificados.

classificação de estudos considerando-se o tipo de contribuição. O tipo de contribuição se refere ao tipo de intervenção sendo estudada e investigada no estudo, que pode ser um processo, um método, um modelo, uma ferramenta, uma métrica etc. Nesta RSL, o tipo de contribuição foi analisado e as definições utilizadas para cada tipo de contribuição foram feitas utilizando-se o documento *ISO/IEC/IEEE International Standard - Systems and Software Engineering Vocabulary* (ISO/IEC/IEEE, 2017). Este documento é uma terminologia que fornece um vocabulário comum aplicável a sistemas e trabalhos na área de Engenharia de Software. Para a classificação em questão, foram identificados trabalhos discutindo: arquiteturas, algoritmos, frameworks, ferramentas, middleware, plataforma, e protocolo. No padrão da ISO/IEC/IEEE (2017), as definições para os tipos de contribuição encontrados, traduzidos pela autora deste documento, são:

> "Arquitetura 1. Conceitos ou propriedades fundamentais de um sistema em seu ambiente incorporados em seus elementos, relacionamentos e nos princípios de seu projeto e evolução; 2. Conjunto de regras para definir a estrutura de um sistema e as inter-relações entre suas partes ((ISO/IEC/IEEE, 2017), p. 26).
>
> Algoritmo 1. Conjunto finito de regras bem definidas para a solução de um problema em um número finito de etapas 2. Sequência de operações para realizar uma tarefa específica 3. Conjunto finito ordenado de regras bem definidas para a solução de um problema ((ISO/IEC/IEEE, 2017), p. 16).
>
> Framework 1. Design reutilizável (modelos e/ou código) que pode ser refinado (especializado) e estendido para fornecer parte da funcionalidade geral de muitos aplicativos; 2. Subsistema de software parcialmente concluído que pode ser estendido instanciando adequadamente alguns *plugins* específicos ((ISO/IEC/IEEE, 2017), p. 190).
>
> Protocolo 1. conjunto de convenções que governam a interação de processos, dispositivos e outros componentes dentro de um sistema ((ISO/IEC/IEEE, 2017), p. 356)".

Na Tabela 7 são listados os tipos de contribuição (na segunda coluna), o foco da contribuição (na terceira coluna) e o tipo de arquitetura para a qual a contribuição foi projetada ou aplicada (na quarta coluna). Os tipos de contribuição identificados foram fundamentais para responder a QP3 (Quão madura é a pesquisa sobre tolerância a defeitos para Aplicações para Cidades Inteligentes?).

3.5.4 Domínios e Subdomínios de Aplicação para Aplicações para Cidades Inteligentes

Essa classificação foi efetuada de acordo com a classificação estabelecida no estudo de Sãnchez-Corcuera et al. (2019). O trabalho sugere quatro domínios de aplicação para cidades inteligentes, de acordo com os serviços oferecidos na cidade e os componentes de sua infraestrutura. O estudo apresenta também um subconjunto de seus sub-domínios associados. Na classificação dos estudos selecionados na RSL, consideraram-se tais domínios e subdomínios para categorizar os tipos de Aplicações para Cidades Inteligentes, sendo eles: domínio relacionado ao governo (*Government-related domain*), domínio relacionado aos cidadãos (*Citizen-related domain*), domínio relacionado aos negócios (*Business-related domain*) e domínio relacionado ao ambiente (*Environment-related domain*). Além de tais domínios, um domínio chamado Genérico foi associado, o que significa que a solução é genérica e pode ser aplicada a mais de um domínio. Na Tabela 9 listam-se os domínios e subdomínios de aplicação abordados nos estudos selecionados.

A maioria dos estudos foi classificada como domínio Genérico. Foram identificados três estudos com mais de um subdomínio sendo abordado, ou seja, diferentes aplicações com diferentes alvos. Os domínios explícitos mais recorrentes são os relacionados ao

Estudo	Tipo de Contribuição	Foco da Contribuição	Tipo de Arquitetura
(KHAN et al., 2021)	Algoritmo	Algoritmo para Agregação de Dados	Fog/Cloud
(MOHAMED; AL-JAROODI; JAWHAR, 2019)	Algoritmo	Algoritmo para Tolerância a Defeitos	Fog
(HASEBE et al., 2017)	Algoritmo	Algoritmo para Tolerância a Defeitos	Não especificado
(SASU; PUIU; NECHIFOR, 2016)	Algoritmo	Algoritmo para Tolerância a Defeitos	Não especificado
(SHAMSI, 2020)	Algoritmo	Algoritmo para Tolerância a Defeitos	Não especificado
(COSTA; NASSAR; DANTAS, 2022)	Algoritmo	Algoritmo para Listagem e Escolha de Sensores	Fog
(FERRARI et al., 2022)	Algoritmo	Algoritmo para Tolerância a Defeitos	Cloud
(JESUS et al., 2022)	Algoritmo	Algoritmo para Reconfiguração de Sensores	Não especificado
(MANOGARAN; NGUYEN, 2022)	Algoritmo	Algoritmo para Melhorar a Troca da Qualidade de Dados	Não especificado
(PASTóRIO; CAMARGO, 2021)	Algoritmo	Algoritmo para Geolocalização	Cloud
(TANCEV; TORO, 2021)	Algoritmo	Algoritmo para Correção de Sensores	Não especificado
(WANG; LIU; CHOO, 2021)	Algoritmo	Algoritmo para Agregação de Dados	Edge/Cloud (necessidade de adaptação)
(YANG; PARK; KIM, 2022)	Algoritmo	Algoritmo para Aquisição de Dados	Cloud
(YU et al., 2022a)	Algoritmo	Algoritmo para Controle de Formação	Não especificado
(YUAN et al., 2021)	Algoritmo	Algoritmo para Estimação de Dados	Não especificado
(HE et al., 2016)	Arquitetura	Arquitetura de Rede	Edge
(MODARRESI; SYMONS, 2020)	Arquitetura	Arquitetura de Rede	Edge
(ALKADY et al., 2019)	Arquitetura	Arquitetura de Rede	Fog
(LIU et al., 2017)	Arquitetura	Arquitetura para Coleta de Dados	Cloud
(ABREU et al., 2017)	Arquitetura	Arquitetura para Coleta de Dados	Cloud e Cloudlets (Nós Fog)
(MORA-MORA et al., 2015)	Arquitetura	Arquitetura para Coleta de Dados	Cloud
(HAKIRI; GOKHALE; BERTHOU, 2019)	Arquitetura	Arquitetura de Rede	Edge
(POWAR et al., 2019)	Arquitetura	Arquitetura de Rede	Não especificado
(SHAH et al., 2019)	Arquitetura	Arquitetura para Gerenciamento de Desastres	Cloud
(BAWANY; SHAMSI, 2019)	Framework	Framework para Segurança	Não especificado
(ALZOMAN; ALENAZI, 2020)	Arquitetura	Arquitetura de Rede	Cloud
(AHMAD et al., 2021)	Arquitetura	Arquitetura de Software como um Serviço	Edge
(ANUPONG et al., 2022)	Arquitetura	Arquitetura de Rede	Não especificado
(BIDI; MOVAHEDI; MOVAHEDI, 2021)	Arquitetura	Arquitetura baseada em Fog Computing	Fog
(KHAN; KIM; PARK, 2022)	Arquitetura	Arquitetura de Sistema de Monitoramento de Poluentes	Não especificado
(KIM; BEN-OTHMAN, 2021)	Arquitetura	Arquitetura de Sistema de Barreira Emocional Virtual Assistida	Não especificado
(KURESHI et al., 2021)	Arquitetura	Arquitetura de Sistema de Monitoramento da Qualidade do Ar	Não especificado
(MIRZA; GEORGAKOPOULOS; YAVARI, 2023)	Arquitetura	Arquitetura de Plataforma para Cidades Inteligentes	Cloud
(NANDURY; BEGUM, 2015)	Arquitetura	Arquitetura de Plataforma para Cidades Inteligentes	Cloud
(PUIU et al., 2016)	Framework	Framework para Coleta e Processamento de Dados	Cloud
(KHARCHENKO et al., 2022)	Framework	Framework para Tolerância a Defeitos	Não especificado
(HAMDAOUI et al., 2020)	Protocolo	Protocolo de Roteamento	Edge/Cloud
(ALI et al., 2022)	Protocolo	Protocolo de Comunicação	Não especificado
(ALJOHANI; ALENAZI, 2021)	Protocolo	Protocolo de Roteamento	Não especificado
(HASEEB et al., 2022)	Protocolo	Protocolo de Roteamento	Não especificado
(MOHAPATRA; RATH, 2021)	Protocolo	Protocolo de Roteamento	Não especificado
(ALI; ZAFAR, 2023)	Protocolo	Protocolo de Comunicação	Cloud

Tabela 8 – Tipo de contribuição e tipo de arquitetura identificada nos estudos.

Meio Ambiente e relacionados aos Cidadãos. Considerando todos os subdomínios, Smart Grid foi o subdomínio mais mencionado, seguido de Smart Traffic. Simultaneamente, o domínio relacionado a Negócios foi raramente abordado (abordado uma vez), sendo que alguns de seus subdomínios não abordados; são eles: Publicidade (uso de tecnologia para publicidade inteligente), Agricultura (garantindo a segurança alimentar e agricultura inteligente), e Logística (apoio à organização e implementação de processos logísticos complexos). O domínio relacionado ao Governo foi o segundo menos abordado e não foram encontrados subdomínios de Governo Eletrônico (fornecimento de serviços usando ferramentas e tecnologia) e Governo Transparente. Outros subdomínios relevantes que não receberam muita atenção, com apenas um estudo de cada, são Congestionamento de Tráfego, Resposta a Emergências e Gestão da Água. De acordo com Sãnchez-Corcuera et al. (2019), o Congestionamento de Tráfego não é apenas um problema econômico, tendo em vista que também prejudica a qualidade de vida dos cidadãos. Além disso, aplicações focadas em Resposta a Emergências tentam reduzir efeitos de riscos urbanos, como desastres naturais, salvando vidas de cidadãos. A Gestão da Água, ainda dentro do domínio relacionado ao Governo, tenta monitorar o consumo de água e a qualidade da água, impactando a saúde dos cidadãos e em questões ambientais. Portanto, torna-se evidente a importância de empregar esforços para desenvolver aplicações confiáveis para subdomínios que têm um impacto direto na vida dos cidadãos.

3.5.5 Tipos de Arquitetura

Na Tabela 8, na última coluna, lista-se o tipo de arquitetura de sistema ao qual a solução apresentada pelo estudo foi proposta para ser encaixada. No geral, foram encontrados estudos que investigam soluções para Computação em Nuvem (*do inglês, Cloud Computing*), Computação em Névoa (*do inglês, Fog Computing*) e Computação de Borda (*do inglês, Edge Computing*), bem como soluções híbridas. Dessa forma, foram identificados: 13 estudos que exploraram arquiteturas baseadas em nuvem, 4 estudos exploraram arquiteturas baseadas em borda, 4 estudos exploraram arquiteturas baseadas em névoa, e outros estudos que exploraram arquiteturas híbridas (2 estudos baseados em névoa/nuvem e 2 estudos para baseados em borda/nuvem). Dentre os estudos selecionados, 19 deles não especificaram uma arquitetura alvo para a qual a solução foi proposta.

A arquitetura mais recorrente foi a baseada em nuvem. Khan et al. (2021) mencionam que problemas como conectividade, alta latência e sobrecarga são enfrentados por *smart grids* (Redes Elétricas Inteligentes) baseadas em nuvem, que fazem parte do domínio Aplicações para Cidades Inteligentes. Conforme mencionado por Mohamed, Al-Jaroodi e Jawhar (2019), as limitações da computação em nuvem podem ser superadas pela computação em névoa. Segundo os autores, ao utilizar serviços de computação operando próximos a dispositivos IoT, ou seja, nós *fog*, melhorias podem ser alcançadas em relação à latência, localização, mobilidade, e melhoria de serviços de transmissão contínua.

Domínio	**Subdomínio**	**# de estudos**	**Estudos**
Domínio Relacionado aos Cidadãos	Tráfego Inteligente	3	(HAKIRI; GOKHALE; BERTHOU, 2019; ALKADY et al., 2019; MANOGARAN; NGUYEN, 2022)
	Transporte Público	2	(LIU et al., 2017; HASEBE et al., 2017)
	Assistência Médica	2	(KHARCHENKO et al., 2022; KIM; BEN-OTHMAN, 2021)
	Congestionamento de Tráfego	1	(YUAN et al., 2021)
Domínio Relacionado ao Governo	Resposta a Emergências	1	(SHAH et al., 2019)
	Monitoramento da Cidade	2	(ABREU et al., 2017; AL-JOHANI; ALENAZI, 2021)
Domínio Relacionado ao Ambiente	Redes Elétricas Inteligentes	5	(ANUPONG et al., 2022; KHAN et al., 2021; HE et al., 2016; MOHAPATRA; RATH, 2021; WANG; LIU; CHOO, 2021)
	Controle de Poluição	3	(KHAN; KIM; PARK, 2022; KURESHI et al., 2021; TANCEV; TORO, 2021)
	Habitação Inteligente	1	(MODARRESI; SYMONS, 2020)
	Gestão Inteligente de Água	1	(POWAR et al., 2019)
Domínio Relacionado a Negócios	Gestão Empresarial	1	(MORA-MORA et al., 2015)
(Múltiplos domínios) Domínios Relacionados ao Governo e ao Ambiente	Segurança Pública, Serviço Público, Controle de Poluição e Monitoramento da Cidade	1	(ALZOMAN; ALENAZI, 2020)
(Múltiplos domínios) Domínios Relacionados aos Cidadãos, ao Governo e ao Ambiente	Tráfego Inteligente, Monitoramento da Cidade e Controle de Poluição	2	(PUIU et al., 2016; SASU; PUIU; NECHIFOR, 2016)
Domínio Genérico	-	18	(SHAMSI, 2020; AHMAD et al., 2021; ALI; ZAFAR, 2023; ALI et al., 2022; COSTA; NASSAR; DANTAS, 2022; FERRARI et al., 2022; HASEEB et al., 2022; BIDI; MOVAHEDI; MOVAHEDI, 2021; JESUS et al., 2022; MIRZA; GEORGAKOPOULOS; YAVARI, 2023; PASTóRIO; CAMARGO, 2021; YANG; PARK; KIM, 2022; YU et al., 2022b; YU et al., 2022a; NANDURY; BEGUM, 2015; HAMDAOUI et al., 2020; BAWANY; SHAMSI, 2019; MOHAMED; AL-JAROODI; JAWHAR, 2019)

Tabela 9 – Domínios e sub-domínios de aplicação dos estudos selecionados.

Costa, Nassar e Dantas (2022) e Ahmad et al. (2021) afirmam que grandes volumes de dados podem ser gerenciados e processados adequadamente pela computação em nuvem. Nesse contexto, é exigido que os nós de rede estejam mais próximos do usuário final para reduzir o tempo entre a solicitação e a resposta, devido à presença de aplicações sensíveis à latência. Alternativamente, Wang, Liu e Choo (2021) argumentam que dados com requisitos relacionados a atraso de entrega dos dados podem ser gerenciados utilizando-se nós de computação em borda, ao mesmo tempo em que a largura de banda também pode ser reduzida. De acordo com Bidi, Movahedi e Movahedi (2021), a computação em nuvem poder apoiar o processamento e armazenamento de serviços de Aplicações para Cidades Inteligentes. Segundo os autores, para diminuir o atraso relacionado à transmissão de dados de/para a nuvem, a proximidade trazida pela computação em névoa pode aumentar a capacidade de resposta dos serviços urbanos. Conforme declarado pelos autores, problemas de conectividade e latência que podem ocorrer em um ambiente de computação em nuvem podem ser melhorados pela computação de borda, enquanto o uso

de um modelo híbrido apoiado pela computação em névoa pode melhorar a capacidade de resposta dos serviços.

3.6 Classificação de Técnicas de Tolerância a Defeitos em Aplicações para Cidades Inteligentes

Técnica	Subtécnica	ID	Descrição da Subtécnica
Reconfiguração de Sistemas (RS)	Reconfiguração de Redes Elétricas inteligentes	RS1	Envolve a adaptação dinâmica do componente da rede inteligente (a rede controladora ou medidores inteligentes).
	Reconfiguração de Rotas	RS2	Envolve a adaptação dinâmica dos caminhos de comunicação em resposta às mudanças na rede.
	Reconfiguração de Sensores	RS3	Envolve a substituição dinâmica do sensor para obter o estado do ambiente.
	Estimação de Dados	RS4	Envolve estimar dinamicamente um valor para substituir aquele considerado defeituoso.
	Reconfiguração de Veículos	RS5	Envolve substituir dinamicamente o veículo que envia dados para a aplicação.
	Reconfiguração de Atuadores	RS6	Envolve a substituição dinâmica do atuador para executar as ações planejadas.
	Reconfiguração de Diferentes Tipos de Componentes	RS7	Envolve substituir dinamicamente a computação do componente que falhou por outro considerado íntegro.
Diversidade (D)	Diversidade de Sensores	D1	Envolve utilizar dinamicamente os resultados de mais de um sensor para obter o valor utilizado pela aplicação.
	Diversidade de Dados	D2	Envolve criar uma réplica ou réplicas do valor dos dados a serem utilizados caso a fonte principal falhe.
	Diversidade de Diferentes Componentes	D3	Envolve a utilização de componentes redundantes que estão sempre ativos e enviam dados ao sistema independente da falha de um componente.
Retentar (R)	Retransmissão de Dados	R1	Envolve o envio de dados para o aplicativo novamente dentro de um intervalo de tempo.

Tabela 10 – Catálogo de tipos de técnicas e subtécnicas de Tolerância a Defeitos identificadas.

Nesta seção descrevem-se os estudos que abordaram técnicas de tolerância a defeitos no contexto de Aplicações para Cidades Inteligentes. Foram identificadas três técnicas no conjunto de estudos selecionados: Diversidade (D), Reconfiguração de Sistema (RS) e Retentar (R). De acordo com as características dos estudos, foi criada uma sub-classificação que inclui um conjunto de *subtécnicas*. Essa sub-classificação é baseada no tipo de estratégia utilizada para recuperar o sistema para que o mesmo passe de um estado incorreto para um estado livre de erros.

Na Tabela 10 são listados os grupos de subtécnicas identificados, bem como suas descrições. Os estudos foram classificados em sete subtécnicas relacionadas à Reconfiguração de Sistema (RS): Reconfiguração de smart grids (RS1), Reconfiguração de Rotas (RS2), Reconfiguração de Sensor (RS3), Estimação de Dados (RS4), Reconfiguração de Veículo (RS5), Reconfiguração de Atuador (RS6) e Reconfiguração de Diferentes Componentes de Software ou/e Hardware (RS7); três subtécnicas são relacionadas à Diversidade (D): Diversidade de Sensores (D1), Diversidade de Dados (D2) e Diversidade de Diferentes Componentes (D3); e uma subtécnica de Retentar (R): Retransmissão de Dados (R1).

Técnica	Subtécnica	ID-Subtécnica	Autores
RS	Reconfiguração de Redes Elétricas Inteligentes	RS1	He et al. (2016), Wang, Liu e Choo (2021)
	Reconfiguração de Rotas	RS2	Hasebe et al. (2017), Bawany e Shamsi (2019), Hakiri, Gokhale e Berthou (2019), Hamdaoui et al. (2020), Modarresi e Symons (2020), Aljohani e Alenazi (2021), Mohapatra e Rath (2021), Ahmad et al. (2021), Anupong et al. (2022), Ferrari et al. (2022), Haseeb et al. (2022)
	Reconfiguração de Sensores	RS3	Mora-Mora et al. (2015), Liu et al. (2017), Abreu et al. (2017), Costa, Nassar e Dantas (2022), Jesus et al. (2022), Khan, Kim e Park (2022), Yang, Park e Kim (2022)
	Estimação de Dados	RS4	Sasu, Puiu e Nechifor (2016), Puiu et al. (2016), Pastório e Camargo (2021), Tancev e Toro (2021), Yuan et al. (2021)
	Reconfiguração de Atuador	RS6	Yu et al. (2022b), Yu et al. (2022a)
	Reconfiguração de Diferentes Tipos de Componentes	RS7	Kim e Ben-Othman (2021)
D	Diversidade de Sensores	D1	Alkady et al. (2019), Kureshi et al. (2021)
	Diversidade de Dados	D2	Powar et al. (2019), Khan et al. (2021), Shah et al. (2019), AlZoman e Alenazi (2020), Mirza, Georgakopoulos e Yavari (2023)
	Diversidade de Diferentes Componentes	D3	Kharchenko et al. (2022)
RS e D	Reconfiguração de Diferentes Componentes e Diversidade de Dados	RS7 e D2	Nandury e Begum (2015)
	Reconfiguração de Rotas e Diversidade de Dados	RS2 e D2	Mohamed, Al-Jaroodi e Jawhar (2019)
	Reconfiguração de Diferentes Componentes e Diversidade de Diferentes Componentes	RS7 e D3	Shamsi (2020)
	Reconfiguração de Veículos e Diversidade de Dados	RS5 e D2	Manogaran e Nguyen (2022)
R	Retransmissão de Dados	R1	Ali et al. (2022), Ali e Zafar (2023)

Tabela 11 – Técnicas de Tolerância a Defeitos identificadas, suas subtécnicas e estudos classificados.

Na Tabela 11 apresentam-se as técnicas e subtécnicas que foram identificadas e os estudos que as abordaram. Os estudos foram agrupados por técnica de tolerância a defeitos e são apresentados em ordem cronológica na tabela. Um maior detalhamento a respeito dos estudos é apresentado na próxima seção.

3.6.1 Reconfiguração de Sistema

Os trabalhos descritos a seguir abordaram apenas as subtécnicas relacionadas à Reconfiguração de Sistema (RS).

❑ **RS1: Reconfiguração de Smart Grids**

- ❑ No trabalho de He et al. (2016), os autores propuseram uma arquitetura de rede de microgrids resiliente. Para garantir a confiabilidade da rede durante desastres que causam apagões e instabilidades, propõe-se que microgrids de suporte recebam alertas para aumentar suas saídas para seus níveis máximos, dessa forma auxiliando na geração de energia.
- ❑ Wang, Liu e Choo (2021) criaram uma abordagem para agregar dados de várias fontes em smart grids. A abordagem foi usada para agregar dados, tais como consumo total de eletricidade, número de usuários e consumo total de eletricidade. A

abordagem detecta o medidor inteligente que falhou e, depois disso, todos os usuários, exceto o usuário do medidor com falha, reenviam seus dados criptografados.

- **RS2: Reconfiguração de Rotas**
  - Hasebe et al. (2017) apresentaram um sistema de transporte de passageiros baseado em veículos autônomos que permite que passageiros cheguem aos seus destinos sem a necessidade de mudar de veículo. Para realizar o agendamento para utilização do veículo, o sistema possui um único servidor central, permitindo realizar todo o gerenciamento de navegação da frota instruindo cada veículo. No entanto, se o servidor central apresentar mau funcionamento, todo o sistema de tráfego poderá ser interrompido. Assim, os autores propuseram um método de atribuição automática de uma rota provisória a cada veículo quando o servidor central falha em determinar a rota.
  - Bawany e Shamsi (2019) *(RS2)* apresentaram um framework, chamado SEAL (SEcure e AgiLe), adaptável e baseado em redes definidas por software para proteger Aplicações para Cidades Inteligentes contra ataques DDoS usando-se balanceamento de carga como uma estratégia inicial de mitigação de ataques DDoS. Além disso, a estrutura garante tolerância a defeitos por meio do uso de múltiplas instâncias do controlador de rede.
  - Hakiri, Gokhale e Berthou (2019) *(RS2)* apresentaram um modelo de rede para Cidades Inteligentes e propuseram replicar controladores na rede (no caso, uma rede definida por software) a fim de eliminar um único ponto de falha. Os autores usaram como exemplo um sistema de semáforo inteligente. O sistema coleta dados de sensores, equipamentos de beira de estrada e câmeras para detectar a presença de veículos, ciclistas e pedestres. O sistema utiliza uma rede baseada em controladores (rede definida por software) e uma rede de malha sem fio para conectar os nós. O estudo mencionou um caso de falha no controlador de rede e propôs controladores distribuídos e redirecionamento do fluxo (por exemplo do roteador 1 para o roteador 4, em caso de falha do link).
  - Hamdaoui et al. (2020) *(RS2)* apresentaram um protocolo que se baseia em uma infraestrutura de comunicação de rede ponto a ponto para permitir comunicação entre dispositivos IoT de maneira distribuída. Os autores propuseram que quando um nó falha, ele pode ser substituído por um serviço semelhante disponível em outro nó da rede e propuseram o uso de blockchain para proteção contra *sybil attacks, bribe attacks, long-Range attacks* e *nothing-at-stake attacks*.
  - Modarresi e Symons (2020) *(RS2)* apresentaram um modelo abstrato de rede doméstica para arquiteturas de casas inteligentes. De acordo com a arquitetura, nós móveis podem se conectar a outros pontos de acesso durante alguma falha de comunicação com seu ponto de acesso nativo.

- Bidi, Movahedi e Movahedi (2021) *(RS2)* propuseram uma arquitetura tolerante a defeitos baseada em computação em névoa para composição de serviço em Aplicações para Cidades Inteligentes. Os autores propuseram três algoritmos de otimização (um algoritmo genético, otimização de enxame de partículas, e colônia artificial de abelhas) para resolver o problema de composição de serviço. A arquitetura proposta distribui a carga entre os nós, aumentando a capacidade de resposta do sistema e trazendo um nível de adaptabilidade em caso de falhas.
- Mohapatra e Rath (2021) *(RS2)* propuseram um esquema de roteamento para smart grids com o objetivo de lidar com falhas na infraestrutura de comunicação. Tal esquema pode garantir que os dados de consumo de energia possam ser coletados e gerenciados sem interrupções. O mecanismo fornece uma rota eficiente se a rota selecionada previamente estiver indisponível. Os resultados da simulação mostraram um melhor desempenho em relação a outros esquemas de roteamento.
- Ahmad et al. (2021) *(RS2)* propuseram uma arquitetura que tem como objetivo oferecer soluções para problemas complexos como um serviço com base na otimização de modelos preditivos e orquestração de tarefas. A otimização do modelo usa um módulo de aprendizado de máquina para transformar determinado problema em soluções. A arquitetura funciona como uma plataforma para implementar diferentes cenários como casas inteligentes, fazendas inteligentes e redes inteligentes. Quando um dos serviços alocados para fazer as tarefas não está acessível no servidor, o próximo solucionador de problemas, ou seja, o próximo serviço, é selecionado na fila.
- Aljohani e Alenazi (2021) propuseram uma estratégia de roteamento multicaminhos para resolver problemas de rede com foco em infraestruturas de rede que apoiam Aplicações para Cidades Inteligentes. O sistema possui dois componentes responsáveis por detectar e recuperar o sistema de erros: (i) o Challenger Detector, que é responsável por detectar uma falha na área de detecção (por exemplo, a presença de fogo ou tempestades), e (ii) o Path Diversity, que é responsável por calcular rotas confiáveis para substituir nós defeituosos. O componente aplica o algoritmo Floyd-Warshall para calcular k caminhos diversos entre nós e nós centrais disponíveis.
- Anupong et al. (2022) *(RS2)* implementaram uma arquitetura de rede para coletar e transmitir dados de consumo de energia em Aplicações para Cidades Inteligentes e lidar com problemas de congestionamento de rede e falhas contínuas de comunicação. A proposta foi um algoritmo de roteamento chamado *Greedy Traffic Routing* que identifica rotas mais confiáveis e otimizadas com base em informações coletadas de nós vizinhos.
- Haseeb et al. (2022) *(RS2)* propuseram um sistema de roteamento supervisionado

que monitora o desempenho da rede e faz ajustes dinâmicos. Quando ocorrem falhas em componentes de rede, o sistema pode ajustar as rotas de comunicação para se recuperar as falhas, garantindo que os dados continuem a ser transmitidos de forma confiável.

- Ferrari et al. (2022) *(RS2)* propuseram técnicas para melhorar a resiliência, flexibilidade e estabilidade de plataformas de IoT. Um protótipo foi construído em uma plataforma comercial de IoT projetada para o setor automotivo. Os autores propuseram usar múltiplos caminhos, o que significa, o uso de diferentes rotas para o tráfego de dados. O objetivo é estimar o atraso na transferência de dados do veículo (ou seja, a fonte de dados) para algum ponto final (ou seja, um nó central). Caminhos de comunicação adicionais são introduzidos quando há caminhos de comunicação menos estáveis considerados inadequados, levando a atrasos de transferência significativamente reduzidos e confiabilidade aprimorada em resultados experimentais.
- Jesus et al. (2022) *(RS2)* apresentaram um algoritmo projetado para planejar dinamicamente o reposicionamento de um nó móvel (ou seja, um dispositivo que coleta dados) em aplicações de sensores distribuídos. Um nó móvel é movido para mais perto desses nós inalcançáveis a fim de restabelecer uma conexão com eles. O estudo apresenta resultados que demonstram como o algoritmo proposto pode ser aplicado para mover o nó móvel pela rede com o objetivo de atender aos requisitos de confiabilidade.

- **RS3: Reconfiguração de Sensor**

  - Mora-Mora et al. (2015) *(RS3)* propuseram um projeto de rede de sensores inteligentes baseada em tecnologias de comunicação RFID (*Radio Frequency Identification*) para Aplicações para Cidades Inteligentes. A solução arquitetural proposta permite a aquisição de dados dos usuários em uma cidade e inclui a substituição de sensores RFID caso alguns deles estejam com defeito.
  - Liu et al. (2017) *(RS3)* propuseram sistemas de sensoriamento denominados *Mobile Sensing Information Gathering Centers (MSIGC)* para ônibus de transporte público com múltiplos sensores (por exemplo, GPS, câmera, sensores de temperatura e umidade) para monitorar a qualidade do ar do veículo, o congestionamento do tráfego nas estradas, e outras condições da cidade. Os autores implementaram sensores redundantes dormentes que são despertados quando os principais estão com defeito.
  - Abreu et al. (2017) *(RS3)* propuseram uma arquitetura para melhorar a resiliência de infraestrutura de IoT com foco em ambientes de Aplicações para Cidades Inteligentes. A arquitetura inclui componente de resiliência que monitora sensores virtuais que serão substituídos em caso de falha.

- ❏ Abreu et al. (2017) *(RS3)* propuseram uma arquitetura para melhorar a resiliência da infraestrutura de IoT com foco em ambientes de Aplicações para Cidades Inteligentes. A arquitetura inclui um componente de resiliência que monitora sensores virtuais, os quais serão substituídos em caso de falha.
- ❏ Costa, Nassar e Dantas (2022) *(RS3)* apresentaram uma abordagem denominada FOCUSeR (*Fog Online Context-Aware Up-to-Date Sensor Ranking*) para classificar sensores. Para detectar anomalias nos sensores e classificá-los, o trabalho apresenta a técnica Matriz de Perfil. Para a técnica de Matriz de Perfil, primeiro divide-se os dados em subsequências menores, isto é, em quartis. Em seguida, calculou-se uma distância (distância Euclidiana) entre cada quartil. Por fim, é criado um perfil de distância que indica quais sequências podem ser de dados são normais. Após isso, uma análise utilizando limiares é utilizada para refinar a análise. A ideia principal é identificar sensores confiáveis próximos que podem ser utilizados como uma alternativa e priorizar fontes de dados com base em critérios como qualidade dos dados.
- ❏ Khan, Kim e Park (2022) *(RS3)* apresentaram um modelo híbrido que combina Redes Neurais Convolucionais e Memória de Curto e Longo Prazo (Redes Neurais Recorrentes) para prever concentrações multivariadas de poluentes do ar. Para prever a concentração de poluentes do ar, a estrutura proposta integra a concentração histórica de poluentes do ar da estação defeituosa com a concentração de poluentes do ar de estações consideradas confiáveis e mais próximas obtidas por meio de uma tabela de distância. Antes de prever a concentração de poluentes, foi realizada uma abordagem para a detecção de anomalias. Para isso, analisou-se valores obtidos menores ou iguais a zero e picos abruptos ou quedas inesperadas nos dados.
- ❏ Yang, Park e Kim (2022) *(RS3)* propuseram um esquema inovador de aquisição de dados urbanos com nome de *Collaborative Reliable Event Transport (cRET).* O esquema depende de sensores alimentados por bateria com módulos Bluetooth de baixo consumo de energia e dispositivos móveis inteligentes que as pessoas carregam em áreas urbanas. Para superar os desafios de comunicação e melhorar a confiabilidade na observação de eventos, o cRET usa uma colaboração baseada na técnica de "escuta" entre sensores. Isso significa que os sensores não apenas transmitem seus dados, mas também respondem a solicitações de outros sensores para melhorar a taxa de entrega de dados.

❏ **RS4: Estimação de Dados**

- ❏ Puiu et al. (2016) *(RS4)* propuseram o framework Citypulse que suporta a criação de serviços Aplicações para Cidades Inteligentes, análise de dados e interpretação de dados de Internet das Coisas (IoT) em larga escala em tempo real. Quando a

qualidade do fluxo de dados é baixa por um longo período de tempo, uma fonte de dados alternativa é selecionada por um componente de adaptação. Sasu, Puiu e Nechifor (2016) *(RS4)* estenderam o trabalho de Puiu et al. (2016) para desenvolver um componente de recuperação de falhas que é integrado à estrutura Citypulse. Os autores propuseram o uso de três algoritmos de aprendizado de máquina: Descida de Gradiente Estocástico para Regressão, Regressor Passivo-Agressivo Online e uma versão personalizada do algoritmo K-Nearest Neighbors para operar dentro do componente. O componente desenvolvido por Sasu, Puiu e Nechifor (2016) é capaz de treinar a si mesmo quando a qualidade do fluxo é alta. Em eventos de baixa qualidade do fluxo, o componente é solicitado a gerar um valor estimado. Estes dois estudos foram classificados no tipo de técnica Reconfiguração do Sistema, pois quando uma fonte de contextual não está disponível, outro conjunto de dados é utilizado.

- Pastório e Camargo (2021) (*RS4*) propuseram melhorar a confiabilidade do rastreamento de objetos em Aplicações para Cidades Inteligentes por meio de uma estratégia de geolocalização híbrida em redes LoRaWAN (*Long Range Wide Area Network*) a fim de determinar a localização dos objetos. A solução consiste em um envio periódico da localização do caminhão para a rede. Em alguns casos em que o GPS não recebe um sinal de satélite, um valor nulo é enviado para indicar que calcular a geolocalização é necessário. A posição do caminhão é estimada usando-se algoritmos de geolocalização (*Received Signal Strength Indicator e Time Difference of Arrival*) e então é enviada para visualização no mapa.

- Tancev e Toro (2021) *(RS4)* propuseram uma solução para recalibrar sensores antigos a fim de fornecer medições confiáveis. Os autores propuseram um método denominado Descida de Gradiente para ser usado com dispositivos móveis confiáveis como referência para ajustar os sensores. Durante algumas interações, sensores e dispositivos móveis compartilharam informações e criaram um conjunto de dados que os pesquisadores usaram para ajustar os sensores. Os nós móveis confiáveis são usados como pontos de referência para ajudar a ajustar os sensores, melhorando o conjunto de dados usado para ajustar os sensores.

- Yuan et al. (2021) *(RS4)* utilizaram um conjunto de técnicas para melhorar a qualidade de dados de GPS de táxis na região de Xian, na China. As etapas seguidas pelos autores foram: (i) Remoção de ruído, que consiste em remover dados ruidosos dos dados originais de GPS; (ii) Redução da dimensionalidade dos dados, na qual os dados são reduzidos numericamente; (iii) Melhoria da qualidade dos dados, em que a qualidade dos dados é melhorada usando medidas como redução da dispersão de dados; e (iv) Correspondência de mapas, que envolve a correspondência de dados de GPS com mapas.

❑ **RS6: Reconfiguração de Atuador**

❑ Yu et al. (2022b) *(RS6)* propuseram um grupo de veículos aéreos não-tripulados para realizar observações ambientais em Aplicações para Cidades Inteligentes para monitoramento de tráfego, coleta de dados ambientais, etc. O foco é garantir que os veículos continuem voando em formação, mesmo que alguma falha aconteça devido a defeitos em um atuador. Para lidar com falhas no atuador e outras incertezas, cada veículo é equipado com uma rede neural. A abordagem foi estendida por Yu et al. (2022a) *(RS6)* com a proposta de um esquema de gerenciamento baseado em aprendizado profundo de reforço multiagente para melhorar a vigilância industrial.

❑ **RS7: Reconfiguração de Diferentes Componentes de Software e/ou Hardware**

❑ Kim e Ben-Othman (2021) (*RS7*) consideraram um cenário de Cidades Inteligentes onde as emoções das pessoas são detectadas por sinais sem fio, como Wi-Fi ou Bluetooth, para serem usadas em uma variedade de aplicativos e serviços de software baseados em emoções (por exemplo, em aplicativos de bem-estar emocional). O estudo propôs o uso de drones (*Unmanned Aerial Vehicles*) para formar uma equipe de resgate para recuperar partes do sensoriamento de emoções que falharam devido a problemas com dispositivos de sensoriamento de IoT.

3.6.2 Diversidade

Os estudos relatados abaixo empregaram apenas a técnica de Diversidade (D).

❑ **D1: Diversidade de Sensores**

❑ Alkady et al. (2019) *(D1)* apresentaram uma arquitetura para Aplicações para Cidades Inteligentes para o monitoramento de ruas urbanas baseando-se na coleta de dados de ruas por meio de sensores tolerantes a defeitos. Um sensor alvo foi triplicado e a saída de cada sensor foi aplicada a um mecanismo de votação. Limiares foram aplicados para a aceitação dos valores de saída dos sensores triplicados. Se houvesse uma grande diferença entre a saída dos sensores triplicados, eles seriam considerados defeituosos.

❑ Kureshi et al. (2021) *(D1)* propuseram soluções baseadas em ciência de dados para selecionar sensores de baixo custo com base em indicadores de qualidade de dados. Em um experimento que envolveu a medição de variantes de poluentes, utilizaram-se três sensores por 48 horas. Os dados foram detectados a cada 15 minutos, e os diferentes padrões de leituras foram analisados. Os sensores foram selecionados após analisar sua consistência por um período de tempo.

❑ **D2: Diversidade de Dados**

- ❑ Khan et al. (2021) *(D2)* propuseram um criptosistema e um mecanismo de autenticação por meio de algoritmo de assinatura digital de curva elíptica para proteger sistemas de redes inteligentes contra injeção de dados falsos e ataques de repetição. Além disso, foi proposto um esquema de agregação de dados, que realiza a agregação de dados a partir de dados coletados de medidores inteligentes. Os autores propuseram o uso de *buffers* (memória de armazenamento temporário), que são usados para armazenar os dados de medição em caso de alguma falha de leitura realizada na rede.
- ❑ Powar et al. (2019) *(D2)* propuseram o uso de diversidade de dados em um sistema de monitoramento de cursos d'água urbanos usando sensoriamento inteligente baseado em Redes de Sensores Sem Fio por meio de um sistema IoT externo com relatórios hidrométricos em tempo real. A arquitetura proposta contém um controlador responsável por coletar e armazenar dados de sensores em caso de falha de comunicação (por exemplo, interrupções de rede ou quedas de pacotes).
- ❑ Shah et al. (2019) *(D2)* propuseram uma arquitetura de referência para gerenciamento de desastres (por exemplo, incêndio em um prédio, bloqueio de estrada devido a qualquer desastre natural, etc) em Aplicações para Cidades Inteligentes que suporta análise em tempo real e offline. Os autores propuseram a realização de backups regulares e armazenamento baseado em nuvem para serem usados caso o sistema primário caia.
- ❑ AlZoman e Alenazi (2020) *(D2)* implementaram um sistema projetado para redes de Aplicações para Cidades Inteligentes. A arquitetura tem um componente responsável por detectar falhas em nós da rede. Para ser resiliente a falhas de links entre os nós, a arquitetura também inclui um componente que fornece caminhos alternativos entre os nós.
- ❑ Mirza, Georgakopoulos e Yavari (2023) *(D2)* propuseram uma plataforma que combina informações de diferentes espaços (cibernético, físico e social) para criar uma compreensão mais completa de contextos tais como Cidades Inteligentes. Os autores implementaram um fator de replicação para manter duas cópias dos dados do sensor em dois *brokers* diferentes (ou seja, os dados do sensor estão replicados em três brokers). Produtores e consumidores ainda podem tolerar que dois *brokers* caiam ao mesmo tempo. No caso de um *broker* se tornar indisponível, outros *brokers* continuam a fornecer dados, garantindo que os dados não sejam perdidos.

3.6.3 Retentar

A seguir são descritos os trabalhos que utilizaram a técnica Retentar (R).

- Ali et al. (2022) *(RT1)* propuseram um modelo matemático e a validação de um protocolo MQTT (*Message Queuing Telemetry Transport*) modificado com o objetivo de melhorar a qualidade de serviço (QoS) em Aplicações para Cidades Inteligentes, garantindo que os dados sejam entregues de forma confiável e em tempo hábil. O modelo matemático foi validado em um ambiente de rede real que simula as condições de comunicação em Aplicações para Cidades Inteligentes.

- Ali e Zafar (2023) *(RT1)* propuseram a implementação do protocolo MQTT (*Message Queuing Telemetry Transport Protocol*) aprimorado com aplicação em cenários de Cidades Inteligentes. O novo MQTT com QoS-0 incorpora regras para retransmitir o conteúdo caso as solicitações não sejam respondidas em um um certo intervalo de tempo.

3.6.4 Reconfiguração de Sistema (RS) e Diversidade (D)

Alguns estudos propuseram mais de uma técnica de Tolerância a Defeitos. São eles:

- Manogaran e Nguyen (2022) *(RS5 and D2)* propuseram um serviço para melhorar a qualidade da troca de dados de Sistemas de Transporte Inteligentes. Isso inclui a troca de dados entre veículos (V2V) e entre veículos e qualquer outro dispositivo (V2X). Os autores propuseram: (i) alterar a conexão quando a conexão com o vizinho for perdida (RS5), (ii) transferir dados para infraestruturas fixas quando as condições não forem boas para transmissão direta (D2) e (iii) priorizar solicitações de dados com base em fatores como a capacidade de conexão de veículos vizinhos e a disponibilidade de largura de banda.

- Nandury e Begum (2015) *(*RS7 and D2) desenvolveram uma arquitetura chamada SWIFT (*Smart-based Infrastructural Framework for Smart Transactions*) para fornecer uma plataforma na qual diferentes dispositivos e objetos presentes em Cidades Inteligentes (por exemplo, sensores, sistemas de controle, veículos, etc) possam interagir. Duas estratégias são propostas: (i) *buffers* de dados para armazenar dados temporariamente (D2); e (ii) transferência de dados no caso de um PE falhar, para que seus dados sejam transferidos para uma unidade de processamento saudável (RS7).

- Shamsi (2020) *(RS7 and D3)* propuseram (i) o uso de redundância dedicada, na qual componentes redundantes são sempre mantidos ativos, independentemente da necessidade imediata (D3); e (ii) redundância ativada apenas sob demanda, na qual o sistema pode ativar a redundância para substituir o componente com falha (RS7).

3.7 Classificação de Defeitos, Erros e Falhas em Aplicações para Cidades Inteligentes

Esta seção descreve defeitos, erros e falhas que foram mencionados como um foco da estratégia proposta para retornar o sistema a um estado saudável. Para esta classificação, considerou-se o termo mencionado pelos autores em relação a defeitos, erros e falhas, ou seja, seguiu-se rigorosamente a descrição do autor. Observa-se que, enquanto alguns autores mencionaram apenas defeitos, outros mencionaram defeitos e consequentes erros e falhas, ou apenas mencionaram falhas. Mesmo em estudos específicos que propuseram ou empregaram uma solução de tolerância a defeitos em Aplicações para Cidades Inteligentes sem executar nenhum experimento, os defeitos, erros ou falhas mencionados como foco das soluções propostas foram coletados.

3.7.1 Defeitos em Aplicações para Cidades Inteligentes

Foram identificados 17 defeitos relatados em 15 estudos. Os identificadores (IDs), as descrições dos defeitos, bem como os estudos que os exploraram estão descritos na Tabela 12. Os defeitos identificados, aqui denominados de Defeitos Específicos (DE), foram agrupados em tipos de Defeitos Genéricos (DG). Na Tabela 13 são listados os tipos de DGs, na qual nas duas primeiras colunas são apresentados o nome e a descrição dos defeitos genéricos, respectivamente. Na última coluna da Tabela 13 são listadas as falhas específicas relacionadas e as subtécnicas que foram exploradas para lidar com os defeitos e as consequentes falhas.

3.7.2 Erros em Aplicações para Cidades Inteligentes

Dos 15 estudos que mencionaram defeitos, 5 deles também relataram erros como consequência dos defeitos ativados. Por exemplo, no caso da ocorrência do defeito DE06, erros em dados de sensores dedicados ao monitoramento de estacionamento e tráfego foram descritos por Puiu et al. (2016) e Sasu, Puiu e Nechifor (2016), indicando que a qualidade dos dados era baixa e não refletia com precisão o ambiente real. Descrições de erros em dados de sensores de temperatura (relacionados ao DE11) também foram relatados por Costa, Nassar e Dantas (2022), assim como erros em dados de sensores de GPS (relacionados ao DE16) relatados por Yuan et al. (2021). Nesses casos, foi diretamente mencionado pelos autores que os dados foram detectados como estando em um estado incorreto, e, a partir dessa detecção, a solução foi então invocada. Kureshi et al. (2021) e Tancev e Toro (2021) discutiram erros relacionados a dados coletados por sensores de gás, embora tais erros não tenham sido associados a uma causa específica (ou seja, um defeito) e, portanto, não foram incluídos na Tabela 12. Por fim, foi observado que todos os erros estavam associados a sensores com objetivos variados.

ID	Descrição do Defeito	Erros
DE1	Um defeito em um medidor inteligente (KHAN et al., 2021)	
DE2	Um defeito em um nó da rede (HAMDAOUI et al., 2020)	
DE3	Um defeito em algum sensor de um conjunto de sensores, tais como sensores de temperatura, umidade, luz, poeira e dióxido de carbono, sensores de umidade do solo e sensores de nível de água (ALKADY et al., 2019)	
DE4	Um defeito em um controlador da rede (BAWANY; SHAMSI, 2019)	
DE5	Um defeito em sensores de monitoramento ambiental e de ruído (ABREU et al., 2017)	
DE6	Defeitos em fontes de informação como as que fornecem dados de estacionamento e tráfego (PUIU et al., 2016; SASU; PUIU; NECHIFOR, 2016)	Dados errôneos de sensores dedicados ao monitoramento de estacionamento e tráfego
DE7	Um defeito em um nó sensor da rede (MORA-MORA et al., 2015)	
DE8	Um defeito em algum ponto da rede, como por exemplo, situações de congestionamento, atraso de comunicação, falhas de roteamento e de link (SHAMSI, 2020)	
DE9	Defeitos de hardware (cabos envelhecidos, comutadores, servidores na nuvem) (SHAMSI, 2020)	
DE10	Defeitos de software que podem levar à degradação do serviço (SHAMSI, 2020)	
DE11	Um defeito em um sensor de temperatura (COSTA; NASSAR; DANTAS, 2022)	Dados errôneos de sensores de temperatura
DE12	Um defeito em componentes físicos de hardware (KHARCHENKO et al., 2022)	
DE13	Defeitos de projeto causados por ações errôneas durante a especificação de requisitos, desenvolvimento ou verificação (KHARCHENKO et al., 2022)	
DE14	Um defeito em um medidor inteligente (WANG; LIU; CHOO, 2021)	
DE15	Um defeito em um atuador que faz mudanças físicas na aeronave, como por exemplo, ajustando os movimentos de suas superfícies de controle (YU et al., 2022b; YU et al., 2022a)	
DE16	Um defeito em um sensor de GPS (YUAN et al., 2021)	Dados errôneos do sensor de GPS
DE17	Um defeito em sensores de GPS, câmera, temperatura e umidade (LIU et al., 2017)	

Tabela 12 – Defeitos Específicos (DE) e resultantes erros.

3.7.3 Falhas em Aplicações para Cidades Inteligentes

A análise dos estudos também revelou um conjunto de falhas que afetam Aplicações para Cidades Inteligentes. As falhas e os estudos que as abordam estão listados na Tabela 14. No total, falhas nas quais as aplicações estavam fora do comportamento esperado foram reportadas em 34 estudos, nos quais ações de recuperação foram necessárias. Foram encontradas 34 falhas, aqui denominadas de Falhas Específicas (FEs). Estudos que mencionaram as falhas que variam de FE1 até FE26 não associaram tais falhas a defeitos específicos que as causam. Estudos que mencionaram as falhas FE27 a FE34, por outro lado, associaram defeitos específicos como sendo as causas da falhas abordadas (os identificadores dos defeitos específicos são apresentados entre parênteses, ao lado dos identicadores da falhas específicas).

A partir da lista de FEs, foi elaborada uma lista com 6 tipos de Falhas Genéricas (FGs), apresentadas na Tabela 15. As duas primeiras colunas da tabela apresentam, respectivamente, o nome e a descrição das falhas genéricas. A última coluna lista as FEs relacionadas e, entre parênteses, os IDs das subtécnicas de tolerância a defeitos exploradas para lidar com elas.

Tipo de Defeito	Descrição	DEs Associados e Técnicas Exploradas
DG1-Defeitos em Atuadores	Atuadores podem ser usados em Aplicações para Cidades Inteligentes para converter sinais de controle em ações físicas. Uma falha neste componente pode alterar o comportamento correto dos componentes controlados da aplicação.	DE15-(RS6)
DG2-Defeitos em Sensores	Sensores são usados em Aplicações para Cidades Inteligentes para obter o estado do ambiente que está ao redor dos sensores. Defeitos em sensores podem comprometer o comportamento do aplicativo, uma vez que ele é sensível ao contexto (dependente do contexto). Além disso, os dados apresentados pelos aplicativos podem ser enviesados devido a erros.	DE3-(D1), DE5-(RS3), DE6-(RS4), DE7-(RS3), DE11-(RS3), DE16-(RS4), DE17-(RS3)
DG3-Defeitos em Nós ou em Rotas da Rede	Tais defeitos impedem que os dados fluam corretamente de um ponto para outro na rede. O roteamento incorreto pode causar caminhos interrompidos, rotas que causem alta latência, minimizando o throughput e evitando o uso eficiente dos recursos da rede.	DE2-(RS2), DE4-(RS2), DE8-(RS7 and D3)
DG4-Defeitos em Diferentes Dispositivos de Hardware	Esses defeitos referem-se a qualquer defeito em hardware (exceto sensores ou atuadores) identificados em qualquer etapa do ciclo de vida de execução da aplicação.	DE12-(D3), DE9-(RS7 and D3)
DG5-Defeito em Código ou Projeto	Esses defeitos referem-se a uma defeito de projeto (arquitetura, estrutura ou planejamento) ou de codificação (por exemplo, parâmetro ou expressão errada em uma chamada de função) que gera erros e falhas na aplicação.	DE10-(RS7 and D3), DE13-(D3)
DG6- Defeito em Infraestrutura de Redes Elétricas Inteligentes Fault in a Smart Grid Infrastructure	Esses defeitos referem-se a uma falha na infraestrutura da rede, por exemplo, em um medidor inteligente (isto é, estruturas de medição) ou em centros de coleta que contêm suporte computacional.	DE1-(D2), DE14-(RS1)

Tabela 13 – Lista de tipos de Defeitos Genéricos.

3.8 Respostas às Questões de Pesquisa

A seguir apresenta-se as respostas para as três questões de pesquisa QP1, QP2 e QP3 estabelecidas nesta RSL.

3.8.1 RQ1 - Quais técnicas de tolerância a defeitos têm sido empregadas em Aplicações para Cidades Inteligentes?

Entre várias técnicas investigadas por muitos anos (LEE; ANDERSON, 1990), três bem conhecidas foram exploradas na literatura selecionada sobre Aplicações para Cidades Inteligentes. São elas: Diversidade (D), Reconfiguração do Sistema (RS), e Retentar (R). Entre as subtécnicas, a subtécnica mais explorada – em 15 estudos, no total – foi a Reconfiguração de Rotas (RS2). Essa adaptação dinâmica permite que o sistema se recupere de interrupções ou congestionamento de rede, reduzindo a latência e maximizando a velocidade e a escalabilidade da rede. No entanto, essa adaptação automática pode trazer um desafio de segurança devido aos novos caminhos estabelecidos, além de aumentar a sobrecarga de processamento envolvida no monitoramento da rede.

A segunda subtécnica mais explorada foi Diversidade de Dados DT2, com 8 estudos. Essa técnica pode aumentar a disponibilidade de dados, mas exige mais capacidade de armazenamento, o que pode ser um problema com grandes volumes de dados. A terceira

ID Falha Específica	**Descrição da Falha**
FE1	Uma falha no controlador da rede (HAKIRI; GOKHALE; BERTHOU, 2019)
FE2	Uma falha no servidor central ao tentar alcançar outro servidor (HASEBE et al., 2017)
FE3	Um dos serviços alocados para realizar as tarefas não está acessível no servidor (AHMAD et al., 2021)
FE4	Falha em uma solicitação de dados (serviço indisponível) (ALI; ZAFAR, 2023)
FE5	Falha em uma solicitação de dados (serviço indisponível) (ALI et al., 2022)
FE6	Falha na entrega de dados devido a interrupções na rede causadas por desastres naturais como tempestades ou incêndios, etc. (ALJOHANI; ALENAZI, 2021)
FE7	Falhas de link (ANUPONG et al., 2022)
FE8	Um caminho de comunicação menos estável considerado inadequado (FERRARI et al., 2022)
FE9	Uma falha em componentes da rede (HASEEB et al., 2022)
FE10	Indisponibilidade do serviço em caso de falha em uma camada da rede (BIDI; MOVAHEDI; MOVAHEDI, 2021)
FE11	Falhas em nós ou links, como um componente eletrônico defeituoso ou descarga completa da bateria (no caso de nós), ou interferência de rádio, ruído, canal ocupado ou colisões de dados (no caso de links) (JESUS et al., 2022)
FE12	Uma falha em um dispositivo de IoT dedicado ao monitoramento do ambiente devido ao esgotamento da bateria, travamento ou mau funcionamento devido a diferentes razões, como falha de equipamento, problemas de integração, conectividade ou sobrecarga do dispositivo (KHAN; KIM; PARK, 2022)
FE13	Uma falha em dispositivos de IoT (por exemplo, esses dispositivos IoT podem estimar a emoção dos cidadãos por meio dos procedimentos acordados previamente, incluindo estimativa de batimentos cardíacos e percepção da respiração) (KIM; BEN-OTHMAN, 2021)
FE14	Falha de uma solicitação de dados durante a troca de dados entre veículos (MANOGARAN; NGUYEN, 2022)
FE15	Uma falha em um *service broker* (MIRZA; GEORGAKOPOULOS; YAVARI, 2023)
FE16	Falha na transmissão de dados entre grids ou em um *grid* da rede de *smart grids* (MOHAPATRA; RATH, 2021)
FE17	Uma falha de GPS de um veículo (PASTóRIO; CAMARGO, 2021)
FE18	Uma falha de transmissão entre dois sensores (YANG; PARK; KIM, 2022)
FE19	"... falha completa de alguns nós de neblina devido a razões como falhas de hardware, problemas de conectividade, perda de energia ou ataques físicos/cibernéticos" (MOHAMED; AL-JAROODI; JAWHAR, 2019)
FE20	Uma falha de comunicação entre nós da rede (POWAR et al., 2019)
FE21	Uma falha em dispositivos de hardware (SHAH et al., 2019)
FE22	Uma falha na rede de *smart grid* (HE et al., 2016)
FE23	Uma falha em um nó da rede (ALZOMAN; ALENAZI, 2020)
FE24	Uma falha em dispositivos IoT resultando na indisponibilidade de dados (SHAMSI, 2020)
FE25	Pontos de acesso da rede representam um ponto crítico de falha da estrutura (MODARRESI; SYMONS, 2020)
FE26	Uma falha em elementos de processamento (por exemplo, em dispositivos que recebem ou enviam dados espalhados na cidade) (NANDURY; BEGUM, 2015)
FE27 (DE1)	Falhas em medidores inteligentes que não conseguem enviar seus dados para um determinado período (KHAN et al., 2021)
FE28 (DE2)	Falhas em nós de rede (HAMDAOUI et al., 2020)
FE29 (DE5)	Falhas no monitoramento ambiental e sensores de ruído (ABREU et al., 2017)
FE30 (DE11)	Valores distantes de valores frequentes são considerados falhas (COSTA; NASSAR; DANTAS, 2022)
FE31 (DE12)	Falhas em veículos aéreos não tripulados causadas por falhas em hardware e ataques maliciosos (KHARCHENKO et al., 2022)
FE32 (DE13)	Falhas de software ou firmware de drones (veículos aéreos não tripulados) ou em sua infraestrutura (KHARCHENKO et al., 2022)
FE33 (DE14)	Falhas em *smart grids* que não conseguem enviar dados (WANG; LIU; CHOO, 2021)
FE34 (DE17)	Falhas em sensores (LIU et al., 2017)

Tabela 14 – Falhas Específicas (falhas específicas sem defeitos associados).

subtécnica mais recorrente foi Reconfiguração de Sensores (RS3), que foi explorada em 6 estudos e envolve a mudança dinâmica de um sensor para outro a fim de obter o estado do ambiente. Essa subtécnica pode aumentar a disponibilidade do sistema ao continuar a tarefa de sensoriamento, mesmo em casos de falha do sensor principal. Por outro lado, há um aumento no custo envolvido no uso de hardware adicional e na implementação de estratégias para a seleção do sensor substituto.

Tipo da Falha	**Descrição**	**FEs relacionadas e subtécnicas exploradas**
FG1-Falhas em Nós da Rede ou Falhas em Links/Rotas	Um comportamento inesperado de um nó ou um caminho de rede que apresenta problemas de conexão, perda de pacotes, largura de banda limitada, etc.	FE1-(RS2), FE2-(RS2), FE3-(RS2), FE4-(RS1), FE-(RS1), FE6-(RS2), FE7-(RS2), FE8-(RS2), FE9-(RS2), FE10-(RS2), FE11-(RS2), FE15-(D2), FE18-(RS3), FE19-(RS2 and D2), FE20-(D2), FE23-(D2), SF25-(RS2), FE28-(RS2)
FG2-Falhas em Dispositivos de IoT ou em outros Dispositivos de Hardware	Um comportamento inesperado em um dispositivo de hardware envolvido no ciclo de vida de execução da aplicação.	FE12-(RS3), FE13-(RS7), FE21-(D2), FE24-(RS7 and D3), FE26-(D2 and RS7), FE31-(D3)
FG3-Falhas de Comunicação de Veículos	Um comportamento inesperado do veículo que não consegue se comunicar com outro nó da rede.	FE14-(D2 and RS8), FE17-(RS4)
FG4-Falhas em *Smart Grids*	Um comportamento inesperado em um nó que não envia valores para outro nó.	FE16-(RS2), FE22-(RS1), FE27-(D2), FE33-(RS1)
FG5-Falhas em Sensores	Um valor inesperado é enviado pelo sensor ou o sensor não está respondendo.	FE29-(RS3), FE30-(RS3), FE34-(RS3)
FG6-Falhas em *Firmware*	Um comportamento inesperado em *firmware*.	FE32-(D3)

Tabela 15 – Tipos de Falhas Genéricas, ID da falha associada e ID da subtécnica utilizada.

Outras subtécnicas frequentemente exploradas foram RS7 (Reconfiguração de Diferentes Componentes) e RS4 (Estimação de Dados). A RS7, explorada em 5 estudos, auxilia no aumento da disponibilidade do sistema, continuando a tarefa de sensoriamento em caso de falha do componente principal. No entanto, apresenta a desvantagem de custos adicionais em sua implementação, devido à necessidade de adicionar novos componentes ao sistema. A RS4, por outro lado, foi explorada por 4 estudos e tem a desvantagem de adicionar custos computacionais ao sistema por conta da complexidade e do tempo de resposta envolvidos nos algoritmos de estimação de dados; entretanto, esses algoritmos podem processar e filtrar dados de forma mais precisa, adaptar-se automaticamente a diferentes condições, melhorar sua precisão ao longo do tempo e, em alguns casos, são mais baratos do que adicionar hardware adicional.

De modo geral, as subtécnicas do grupo de Diversidade (D) e Reconfiguração do Sistema (RS) apresentam a desvantagem de custos adicionais (computacionais e financeiros) no projeto, devido à redundância. No caso da Reconfiguração do Sistema, há o desafio adicional da integração de um novo componente de hardware ou software ao sistema. Apesar de a técnica Retentar (R) ser a técnica mais simples de ser implementada, a mesma recebeu pouca atenção; apenas 2 estudos exploraram a subtécnica R1 (Retransmissão de Dados). Uma possível razão para isso está no fato de que esta técnica oferece suporte à recuperação do sistema apenas em falhas transientes (ou seja, são interrupções ou falhas temporárias em um sistema, que desaparecem após um curto período).

3.8.2 RQ2 - Quais são os tipos de defeitos, erros e falhas envolvidos em Aplicações para Cidades Inteligentes, e quais deles foram tratados por técnicas de tolerância a defeitos?

De acordo com o conjunto de estudos analisados, a caracterização de erros no contexto de Aplicações para Cidades Inteligentes é muito limitada. Mais especificamente, apenas 5 estudos forneceram informações muito breves sobre erros nessas aplicações. Portanto, a análise apresentada nesta seção possui como base apenas os defeitos e as falhas identificadas.

Os tipos de defeitos recorrentes na literatura analisada são DG3-Defeitos em Nós ou em Rotas da Rede, e DG2-Defeitos em Sensores. Defeitos desses tipos estão relacionadas a duas características importantes em Aplicações para Cidades Inteligentes: conectividade e sensoriamento. Com relação a DG3, a literatura explora o uso da Reconfiguração de Rotas (RS2) como a principal subtécnica investigada. Defeitos assim podem afetar a conectividade de Aplicações para Cidades Inteligentes, causando atrasos ou perda de dados e, portanto, afetando todo o desempenho da aplicação. Com relação ao tipo DG2. o mesmo pode alterar as ações tomadas pelas aplicações que analisam dados do ambiente. Os resultados indicam que RS3-Reconfiguração de Sensores e RS4-Estimativa de Dados são as principais soluções para lidar com defeitos to tipo DG2. Diversidade (D) (em particular, DG1-Diversidade de Sensores e DG2-Diversidade de Dados) também foi explorada para aprimorar as capacidades de Tolerância a Defeitos em relação a defeitos em sensores.

Outros tipos de defeitos também têm chamado a atenção dos pesquisadores. São os casos de DG1-Defeitos em Atuadores, DG4-Defeitos em Diferentes Dispositivos de Hardware, DG5-Defeitos em Código ou Design, e DG6-Defeitos em Infraestrutura de Smart Grids. Em relação a DG1-Defeitos em Atuadores, a literatura apresenta a Reconfiguração de Sistemas como a única técnica de explorada. A técnica Diversidade não foi identificada como solução para este defeito genérico (por exemplo: dispor de atuadores executando ações em paralelo). Por um lado, destaca-se que o tipo DG1 pode causar consequências menores (por exemplo, consumo ineficiente de energia). Por outro lado, defeitos do tipo DG1 também podem ter consequências maiores, como colocar a vida de cidadãos em risco (por exemplo, um defeito em um atuador que controla um semáforo) ou levar a uma falha completa da operação do sistema controlado. Sobre os tipos DG4-Defeitos em Diferentes Dispositivos de Hardware e DG5-Defeitos em Código ou Design, D3-Diversidade de Diferentes Componentes (hardware e/ou software) foi a principal solução investigada. Reconfiguração de Diferentes Componentes (RS7) também foi explorada em ambos os casos. Finalmente, no caso de DG6-Defeitos na Infraestrutura de Smart Grids, identificaram-se os usos de Diversidade de Dados (D2) e Reconfiguração de Smart Grids (RS1) como as duas únicas soluções até agora. Ressalta-se que DG6 pode comprometer a coleta de dados

na rede inteligente e, portanto, afetar o desempenho de aplicações de software que servem essas redes.

Com relação a falhas, FG1-Falhas em Nós da Rede ou Links/Rotas teve como principal solução a técnica de Reconfiguração de Rotas (RS2). Falhas na rede podem causar interrupções, atrasos ou perdas de pacotes durante a comunicação entre aplicações e outros componentes da cidade. Buscar outro caminho de conectividade pode ser uma boa estratégia para oferecer disponibilidade do serviço em caso de falha. Em relação a falhas do tipo FG2-Failures em dispositivos de IoT e outros dispositivos de hardware, a literatura revela duas subtécnicas exploradas, a saber: Reconfiguração de Sensores (RS3) e Diversidade de Dados (D2). Essas falhas podem afetar sensores, atuadores e outros dispositivos de IoT espalhados pela cidade. Com relação ao tipo FG3-Falhas na Comunicação de Veículos, diferentes subtécnicas foram investigadas: Reconfiguração de Veículos (RS5) e Diversidade de Dados (D2). Esse tipo de falha pode afetar aplicações relacionadas a veículos que comunicam entre si e também com outras aplicações em ambientes urbanos.

Sobre o tipo FG4-Falhas na Infraestrutura de Smart Grids, a literatura indica Reconfiguração de Smart Grids (RS1) e RS2 (Reconfiguração de Rotas) como soluções. Essas falhas afetam a infraestrutura de smart grid, causando interrupções, atrasos ou perdas de dados. Para FG5-Falhas em Sensores, a técnica Reconfiguração de Sensores (RS3) é apresentada como a principal solução. Falhas em sensores podem afetar o comportamento da aplicação, por exemplo, causando sensoriamento de dados imprecisos e trazendo consequências a seus usuários. Finalmente, a respeito de FG6-Falhas em Firmware, a solução proposta foi Diversidade de Diferentes Componentes (RS7). Falhas de firmware podem afetar o controle de diferentes dispositivos de hardware, por exemplo, sensores e atuadores, que são partes importantes de Aplicações para Cidades Inteligentes.

3.8.3 RQ3 - Quão madura é a pesquisa sobre tolerância a defeitos para Aplicações para Cidades Inteligentes?

Na Tabela 16 foram estabelecidas relações entre tipos de contribuição e subtécnicas de tolerância a defeitos. A intenção é lançar alguma luz sobre a probabilidade possibilidade de adoção prática de subtécnicas específicas de tolerância a defeitos em curto prazo. Para isso, foi considerada uma abordagem "madura" aquela que é mais viável para ser usada na prática, em sua versão atual, para desenvolver aplicações tolerantes a defeitos; por exemplo, aquelas que já estão implementadas em um framework ou incorporadas em uma arquitetura. De acordo com os tipos de contribuição observados nos artigos analisados, em ordem, tem-se as contribuições do tipo Framework, Arquitetura, Protocolo e Algoritmo como soluções de mais para menos viáveis de serem utilizadas em aplicações do mundo real.

Subtécnica	**Tipo de contribuição e # de estudos**	**Contribuições não-exploradas**
D1-Diversidade de Sensores (ALKADY et al., 2019; KURESHI et al., 2021)	Arquitetura: 2 estudos;	Algoritmo, Framework, Protocolo
D2-Diversidade de Dados (KHAN et al., 2021; POWAR et al., 2019; SHAH et al., 2019; ALZOMAN; ALENAZI, 2020; MIRZA; GEORGAKOPOULOS; YAVARI, 2023)	Arquitetura: 4 estudos; Algoritmo: 1 estudo;	Framework, Protocolo
D3-Diversidade de Diferentes Componentes do Sistema (KHARCHENKO et al., 2022)	Framework: 1 estudo;	Algoritmo, Arquitetura, Protocolo
RS1-Reconfiguração de Redes Elétricas Inteligentes (WANG; LIU; CHOO, 2021; HE et al., 2016)	Algoritmo: 1 estudo; Arquitetura: 1 estudo;	Framework, Protocolo
RS2-Reconfiguração do Rotas ((HASEBE et al., 2017; FERRARI et al., 2022; HAKIRI; GOKHALE; BERTHOU, 2019; BIDI; MOVAHEDI; MOVAHEDI, 2021; AHMAD et al., 2021; ANUPONG et al., 2022; MODARRESI; SYMONS, 2020; BAWANY; SHAMSI, 2019; HAMDAOUI et al., 2020; MOHAPATRA; RATH, 2021; ALJOHANI; ALENAZI, 2021; HASEEB et al., 2022)	Arquitetura: 5 estudos; Protocolo: 4 estudos; Algoritmo: 1 estudo; Framework: 1 estudo;	-
RS3- Reconfiguração de Sensor (MORA-MORA et al., 2015; COSTA; NASSAR; DANTAS, 2022; JESUS et al., 2022; YANG; PARK; KIM, 2022; KHAN; KIM; PARK, 2022; LIU et al., 2017; ABREU et al., 2017)	Algoritmo: 3 estudos; Arquitetura: 4 estudos;	Framework, Protocolo
RS4-Estimação de Dados (SASU; PUIU; NECHIFOR, 2016; PASTóRIO; CAMARGO, 2021; TANCEV; TORO, 2021; YUAN et al., 2021; PUIU et al., 2016)	Algoritmo: 4 estudos; Framework: 1 estudo;	Arquitetura, Protocolo
RS5-Reconfiguração de Veículo	*Somente em combinação com outras técnicas*	-
RS6-Reconfiguração de Atuador (YU et al., 2022b; YU et al., 2022a)	Algoritmo: 2 estudos	Arquitetura, Framework, Protocolo
RS7-Reconfiguração de Diferentes Componentes (KIM; BEN-OTHMAN, 2021)	Arquitetura: 1 estudo	Algoritmo, Framework, Protocolo
R1-Retransmissão de Dados (ALI et al., 2022; ALI; ZAFAR, 2023)	Protocolo: 2 estudos	Algoritmo, Arquitetura, Framework
RS7-Reconfiguração de Diferentes Componentes e D2-Diversidade de Dados (NANDURY; BEGUM, 2015)	Arquitetura: 1 estudo	Algoritmo, Framework, Protocolo
RS7 - Reconfiguração de Diferentes Componentes de Software e/ou Hardware e D3 - Diversidade de Diferentes Componentes (SHAMSI, 2020)	Algoritmo: 1 estudo	Arquitetura, Framework, Protocolo
RS2-Reconfiguração de Rotas e D2-Diversidade de Dados (MOHAMED; ALJAROODI; JAWHAR, 2019)	Algoritmo: 1 estudo	Arquitetura, Framework, Protocolo
RS5-Reconfiguração de Veículo e D2-Diversidade de Dados (MANOGARAN; NGUYEN, 2022)	Algoritmo: 1 estudo	Arquitetura, Framework, Protocolo

Tabela 16 – Tipos de contribuição e sub-técnicas exploradas.

Análise de contribuições no escopo de arquiteturas: Foi observado que algumas subtécnicas não foram exploradas em uma arquitetura de software definida para Aplicações para Cidades Inteligentes. Estes são os casos de D3-Diversidade de Diferentes Componentes do Sistema, RS4-Estimação de Dados, RS6-Reconfiguração de Atuadores e R1-Retransmissão de Dados. Estes também são os casos de iniciativas de pesquisa que combinam duas subtécnicas como, por exemplo, o uso de RS7-Reconfiguração de Diferentes Componentes e D3-Diversidade de Diferentes Componentes, RS2-Reconfiguração de Rotas e D2-Diversidade de Dados, e RS5-Reconfiguração de Veículos e D2-Diversidade de Dados. Destaca-se a importância das arquiteturas de software para o design e a evolução

do Aplicações para Cidades Inteligentes tolerantes a defeitos. Mais esforços podem ser feitos em relação a essas subtécnicas para descobrir como elas podem ser implementadas em arquiteturas de software para Aplicações para Cidades Inteligentes. Um exemplo é a criação de um componente de software para gerenciar falhas em diferentes componentes de software e hardware, para gerenciar dados errôneos, e gerenciar reconfigurações de atuadores.

Análise de contribuições no escopo de frameworks: Foi observado que muitas subtécnicas não foram implementadas em frameworks. Esses são os casos de D1-Diversidade de Sensores, D2-Diversidade de Dados, RS1-Reconfiguração de Smart Grids, RS3-Reconfiguração de Sensores, RS6-Reconfiguração de Atuadores, RS7-Reconfiguração de Diferentes Componentes, e R1-Retransmissão de Dados. Uma observação semelhante diz respeito ao uso combinado de subtécnicas. Uma possível direção de pesquisa é desenvolver modelos reutilizáveis para facilitar a implementação de técnicas de tolerância a defeitos em código reutilizável a ser estendido por Aplicações para Cidades Inteligentes.

Observou-se que a sumarização de resultados realizada pela resposta à QP3 não foi realizada previamente por estudos primários ou secundários analisados nesta SRL.

3.9 Discussões Adicionais

Um problema relevante no contexto a Cidades Inteligentes está relacionado ao tempo necessário para recuperar Aplicações para Cidades Inteligentes de falhas que podem afetar uma pequena ou grande área de uma cidade. Um subconjunto de estudos selecionados nesta RSL relatou experimentos, particularmente aqueles classificados como pesquisa de validação ou avaliação. Tais estudos foram analisados, com foco em métricas relacionadas ao tempo de recuperação, ou seja, o tempo necessário desde o momento em que um erro é detectado até o momento em que o sistema pode voltar a estar disponível. Destaca-se que as informações sobre o tempo de recuperação, quando disponíveis, podem orientar pesquisas futuras sobre como avaliar e otimizar abordagens de tolerância a defeitos no contexto de Aplicações para Cidades Inteligentes. Assim, a discussão apresentada nesta seção envolve estudos que realizaram medições relacionadas ao tempo de recuperação em cenários com e sem defeitos ativos.

Foram consideradas abordagens avaliadas por uma ou mais métricas que estão associadas à recuperação do sistema, considerando cenários com e sem defeitos ativos. Dito isso, cada estudo possui um contexto muito específico, o que inclui um arranjo experimental feito de forma personalizada sem seguir um padrão único. Portanto, os estudos têm muitas especificidades quanto às métricas que podem refletir o tempo de recuperação das aplicações. Essas configurações experimentais heterogêneas dificultam qualquer tipo de comparação entre as abordagens. A seguir são apresentadas os estudos que mencionam as métricas utilizadas para medir o tempo de recuperação do sistema após a detecção do

erro e aplicação da técnica de recuperação.

Sasu, Puiu e Nechifor (2016) (RS4) calcularam o tempo de treinamento e teste para diferentes conjuntos de dados para técnicas usadas para estimar dados. As técnicas com melhor tempo de treinamento foram Descida de Gradiente Estocástico para Regressão e K-Vizinhos Mais Próximos Incrementais, com valores variando de 0,001 segundos a 0,060 segundos de treinamento para diferentes configurações de parâmetros. Khan et al. (2021) (D2) calcularam o custo de agregação (isto é, tempo para enviar dados de uma microrrede para o nó central) em milissegundos, com base em diferentes porcentagens de nós com falha e variando o número de nós disponíveis. O maior custo de agregação foi menor que 300 milissegundos.

Ahmad et al. (2021) (RS2) apresentaram, por meio de gráficos, o tempo de resposta em milissegundos de acordo com nós da rede falhando simultaneamente. Os autores também avaliaram a escalabilidade com base no *throughput* e na resposta ao longo do tempo, considerando diferentes módulos da arquitetura. Aljohani e Alenazi (2021) (RS2) avaliaram o sistema por meio da simulação da operação normal e situações de desastres como, por exemplo, tempestades e incêndios. Algumas métricas foram analisadas:; por exemplo, tempo de execução, *throughput*, *goodput*, *overhead* e atraso de ponta a ponta. Os experimentos consideraram diferentes tamanhos de rede e números de rotas. Os autores mencionaram que selecionar um valor maior para o número de rotas aumenta o *overhead* em até 200%.

Costa, Nassar e Dantas (2022) (RS3) calcularam que a latência média adicionada pela respectiva abordagem para classificação de sensores é de aproximadamente 0,0016 milissegundos por transação. Quando falhas ou anomalias nas leituras foram identificadas, o algoritmo proposto buscou dados normais de sensores vizinhos. A latência média adicionada pela abordagem foi de aproximadamente 0,225 milissegundos por transação. Jesus et al. (2022) (RS2) apresentaram uma comparação entre a disponibilidade da aplicação com nós coletores estáticos e móveis, o tempo médio de reparo e o ganho médio de disponibilidade. O ganho médio de disponibilidade entre as simulações usando nós de suporte foi de 2,26%, portanto mostrando uma melhoria de confiabilidade. Manogaran e Nguyen (2022), (RS5 e D2), por sua vez, avaliaram a perda de requisições, o tempo de erro com relação às requisições tratadas, a taxa de reconexão de links, o atraso, a taxa média de entrega variando a velocidade e a densidade do veículo, e a sobrecarga de comunicação versus a velocidade do veículo. Um dos resultados indicou que para 100 veículos e 519 requisições, o aplicativo permaneceu 11,06 milissegundos em erro devido a perdas de requisições.

3.10 Direções de Pesquisa

Em conclusão, observa-se que algumas técnicas bastante conhecidas na literatura de dependabilidade e tolerância a defeitos, como Reconfiguração de Sistema e Diversidade,

foram exploradas para recuperar Aplicações para Cidades Inteligentes de diferentes tipos de defeitos, erros e falhas. Por outro lado, as evidências sobre a eficácia das soluções de tolerância a defeitos propostas para o domínio das Cidades Inteligentes ainda são muito limitadas. Com base nas conclusões gerais do estudo, no restante desta seção são discutidos detalhes das limitações da pesquisa atual sobre tolerância a defeitos em Aplicações para Cidades Inteligentes e são apontadas direções de pesquisas futuras neste campo. A discussão é agrupada em quatro tópicos: (i) técnicas utilizadas, (ii) caracterização de defeitos, erros e falhas; (iii) domínios e subdomínios de Aplicações para Cidades Inteligentes; e (iv) tipos de pesquisa realizados e limitações gerais de pesquisa.

(i) Tolerância a Defeitos em Aplicações para Cidades Inteligentes: Nesta RSL identificaram-se três técnicas de tolerância a defeitos, a saber: Diversidade, Reconfiguração do Sistema e Retentar, que foram investigadas para o desenvolvimento de Aplicações para Cidades Inteligentes. No entanto, uma análise mais ampla da literatura relacionada revela que algumas técnicas foram investigadas para outros domínios que estão intimamente relacionados ao de Cidades Inteligentes (Eleuterio et al., 2016; SOUZA; FERRARI, 2022; ZHOU et al., 2019). Por exemplo, não foram identificados estudos que explorassem técnicas de recuperação de erros, como a recuperação de erros por retorno usando pontos de verificação (*checkpoints*) (LEE; ANDERSON, 1990). Também não foram identificadas as técnicas de Ações Atômicas e Ações Atômicas Coordenadas para as fases de confinamento e avaliação de erros (XU et al., 1995). Essas abordagens ajudam a conter a disseminação de erros e minimizar consequências negativas, o que pode ser importante para Aplicações para Cidades Inteligentes que são sensíveis ao contexto e, como na maioria dos casos, distribuídas. Mesmo tendo sido identificada a técnica Diversidade, nenhum estudo abordou as técnicas bem conhecidas de Blocos de Recuperação (*(Recovery Blocks)* (HORNING et al., 1974) ou Programação N-Versões (AVIZIENIS, 1985). Outras fases de tolerância a defeitos, incluindo detecção de erros, recuperação de erros e tratamento de defeitos, são cobertas pelos estudos selecionados. Uma subtécnica que merece atenção é a Estimação de Dados (RS4). Técnicas como Redes Neurais Recorrentes (RNNs), Redes Neurais Convolucionais (CNNs) e Filtros de Kalman podem ser empregadas para estimar dados a partir da análise de valores históricos, contribuindo para a melhoria da precisão e da disponibilidade dos dados. Outras técnicas de Aprendizado de Máquina para análise de dados podem ser exploradas para detectar erros e ajudar a colocar as Aplicações para Cidades Inteligentes em um estado livre de erros. Essas técnicas têm o potencial de ajudar a analisar dados sequenciais de sensores.

(ii) Caracterização de Defeitos, Erros e Falhas em Aplicações para Cidades Inteligentes: Defeitos em nós de rede ou links/rotas foram uma grande preocupação nos estudos selecionados. Esse tipo de defeito é muito importante de ser amenizado devido à alta conectividade nessas aplicações. Defeitos em sensores foram a outra grande preocupação dos pesquisadores.

É muito importante mitigar tais defeitos, pois valores incorretos podem levar aplicações a tomar decisões erradas, o que pode colocar os cidadãos em risco dependendo da criticidade da aplicação. Por exemplo, decisões ruins feitas por aplicações de saúde ou transporte podem causar perda de confiança no sistema por parte dos usuários. Uma lista de defeitos e falhas ajudará os profissionais a priorizar quais partes do sistema devem receber mais atenção ao projetar e implementar soluções de tolerância a defeitos. A evolução da lista de defeitos, erros e falhas que foram apresentadas na síntese desta RSL, em direção a uma lista abrangente semelhante ao que é apresentado no glossário CWE (Common Weaknesses Enumeration) de fraquezas de software[4] pode ajudar os projetistas de Aplicações para Cidades Inteligentes a decidir quais técnicas usar, e quando usá-las.

(iii) Domínios e Subdomínios de Aplicações para Cidades Inteligentes: De acordo com o conjunto de estudos selecionados, os domínios mais explorados foram relacionados ao meio ambiente (10 estudos) e relacionados ao cidadão (8 estudos). Estudos envolvendo tolerância a defeitos para subdomínios relacionados ao governo e aos negócios, por outro lado, são escassos. Enquanto apenas cinco estudos estejam no contexto de subdomínios relacionados ao governo, subdomínios relacionados aos negócios foram abordados em um único estudo. Isso pode ser uma indicação de que poucas aplicações tolerantes a defeitos estão dando suporte a problemas de gerenciamento de cidades (particularmente, governo eletrônico e governo transparente), assim como problemas de gerenciamento de questões econômicas da cidade, em setores como agricultura, empreendedorismo, gestão empresarial e logística. Como tal, esses subdomínios ainda requerem atenção dos pesquisadores quanto à aplicação de tolerância a defeitos e a caracterização de tipos de defeitos, erros e falhas.

(iv) Tipo de Pesquisa e Limitações Gerais da Literatura Analisada: Apenas cinco de 43 estudos que foram analisados nesta RSL foram classificados como pesquisa de avaliação. De acordo com Wieringa et al. (2006), a pesquisa de avaliação requer que a solução proposta seja aplicada na prática (por exemplo, usando a solução em uma cidade real) para permitir uma avaliação empírica e a produção de evidências confiáveis da eficácia da solução no mundo real. Nesta RSL, o pequeno número de estudos que realizaram esse tipo de pesquisa indica que é desafiador para profissionais e pesquisadores avaliar técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes do mundo real. É imperativo que estudos futuros melhorem a robustez dos experimentos, revelando resultados que podem ser aplicados mais diretamente no estado da prática.

3.10.1 Ameaças à Validade

Zhou et al. (2016) estabeleceram um conjunto de ameaças à validade para revisões sistemáticas de literatura na área de Engenharia de Software. Algumas dessas ameaças

[4] <https://cwe.mitre.org/documents/glossary/index.html> - acessado em novembro de 2024.

foram identificadas nesta RSL e estão associadas às fases de planejamento, execução e relato, a saber:

- **Fase de Planejamento** - *(i) Método de busca inapropriado:* Essa ameaça ocorre quando métodos de busca inapropriados são aplicados na recuperação de estudos. Nesta RSL, foi utilizado apenas o mecanismo de busca automática na Scopus. Para mitigar essa ameaça, foram realizadas buscas manuais (*snowballing backward* e *forward*) para aumentar o conjunto de estudos analisados. Em resumo, seguiu-se uma estratégia híbrida (ou seja, uma busca automática com buscas manuais) para recuperar estudos conforme recomendado por outros pesquisadores. Por exemplo, no estudo de Mourao et al. (2017) concluiu-se que usar uma estratégia de busca híbrida ajuda pesquisadores a lidar com desafios enfrentados com a adoção de muitas bibliotecas digitais durante a busca automática. Uma estratégia híbrida envolve utilizar uma biblioteca digital suficientemente representativa (por exemplo, a Scopus, que abrange muitas disciplinas e é definida como um banco de dados de resumos e citações de literatura revisada por pares) e uma estratégia manual como *snowballing*. No conjunto final de estudos selecionados, estudos de diferentes bases indexadas como, por exemplo, IEEE, ACM e ScienceDirect, foram identificados, conforme é exibido nas Tabelas 4 e 5.

- **Fase de Condução** -*(i) Viés na seleção do estudo:* Essa ameaça ocorre quando algum viés é adicionado durante o processo de seleção do estudo. Nesta RSL, a seleção de artigos foi realizada principalmente por um pesquisador. Porém, quando persistiram dúvidas a respeito da inclusão, os critérios de inclusão e exclusão foram discutidos entre dois ou mais pesquisadores envolvidos, que ajudaram a decidir sobre a inclusão de alguns estudos. *(ii) Classificação incorreta do estudo primário:* Tal ameaça ocorre quando o revisor interpreta incorretamente um estudo e, portanto, realiza uma classificação errônea do estudo. Nesta RSL, alguns estudos descreveram a abordagem de tolerância a defeitos e os resultados obtidos em um alto nível de detalhes, o que requer compreensão adequada para obter classificações precisas. Para mitigar classificações errôneas, todos os dados extraídos e classificações de estudos subsequentes foram verificados pelo menos duas vezes por um dos pesquisadores envolvidos.

- **Fase de Relato** - *(i) Falta de avaliação de especialistas*: a falta de experiência pode causar algum viés ou má interpretação na conclusão dos resultados. Para mitigar essa ameaça, os processos de extração e classificação de informações, bem como o processo de relato dos resultados, foram monitorados e revisados por pesquisadores experientes.

3.11 RSL sobre Tolerância a Defeitos em Sistemas Adaptativos e Sistemas Sensíveis a Contexto

Em um estudo secundário que precedeu a RSL descrita neste capítulo, também liderado pela autora deste documento, caracteriza o estado da arte em tolerância a defeitos para SAs e SSCs. Como resultado, a RSL foi publicada no Proceedings of the 3^{rd} IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS) (SOUZA; FERRARI, 2022).

Para responder a questão de pesquisa desta RSL "Quais técnicas de tolerância a defeitos foram propostas para ou aplicadas em SAs e SSCs?", coletou-se dados sobre as técnicas mencionadas em cada estudo selecionado na revisão. Os 41 estudos selecionados foram agrupados com base nas características das técnicas utilizadas para tolerar defeitos. Foram encontradas técnicas relacionadas a Recuperação de Erros, Diversidade, Reconfiguração de Sistema, Retentar, um conjunto denominado Híbridas (envolve o uso de duas ou mais técnicas), um conjunto de técnicas denominado Outras e um conjunto de estudos que abordam Tratamento de Exceções. Uma técnica Híbrida envolve o uso de duas ou mais técnicas. Algumas técnicas que não se encaixaram nos mecanismos das técnicas mencionadas anteriormente foram classificadas como Outras. Na Tabela 17 são apresentadas a quantidade de estudos por técnica bem como o tipo de domínio de aplicação de cada estudo. Observa-se que Diversidade, Reconfiguração de Sistema e Retentar são técnicas que também aparecem como técnicas utilizadas em Aplicações para Cidades Inteligentes identificadas na segunda RSL.

Analogamente, para responder segunda questão de pesquisa: 'Quais defeitos têm sido identificados em SAs e SSCs?", defeitos descritos nos estudos foram analisados. A lista de defeitos identificada nesta RSL é apresentada na Tabela 18. Observa-se que alguns defeitos tais como defeitos em sensores e atuadores aparecem como tipos de defeitos tanto de SAs e SSCs quanto em Aplicações para Cidades Inteligentes.

Os resultados obtidos nesta primeira RSL, auxiliaram o desenvolvimento da classificação de estudos da segunda RSL sobre tolerância a defeitos em Aplicações para Cidades Inteligentes e no entendimento dessas aplicações.

3.12 Considerações Finais

Dada a necessidade de garantir tolerância a defeitos em Aplicações para Cidades Inteligentes, foi conduzida uma RSL para entender a respeito de técnicas de tolerância a defeitos utilizadas e os tipos de defeitos, erros e falhas observados em tais aplicações. Para isso, foi elaborado um protocolo para a RSL que foi conduzida conforme diretrizes bem estabelecidas. Ao final, obteve-se um conjunto de técnicas, defeitos, erros e falhas para tais aplicações. Com o panorama geral obtido, observou-se que algumas técnicas bem estabelecidas na literatura de tolerância a defeitos tem sido utilizadas, enquanto outras,

Tabela 17 – Domínios de aplicação dos estudos e técnicas usadas em SAs e .

Técnica	**Domínio de aplicação**
T1 - Recuperação de erros (6 estudos)	**Aplicação web - web services** (De Lemos, 2006; VARGAS-SANTIAGO et al., 2018) **Leitor de música sensível ao contexto; Sistema de informação de monitoramento de pacientes ciente de contexto; etc** (KULKARNI; TRIPATHI, 2010) **Sistema de transporte inteligente** (CUBO; CANAL; PIMENTEL, 2011) **Sistema de gestão de armazém** (CHEN; YE; JACOBSEN, 2011) **Carro autônomo** (Xu et al., 2013)
T2 - Diversidade (3 estudos)	**Aplicação web - web services** (Tohma, 2004) **Sistema de aeronaves não tripuladas** (CHEN; YE; JACOBSEN, 2011) **Sistema de monitoramento de pacientes** (ZAITER; HACINI, 2018)
T3 - R. de Sistema (14 estudos)	**Redes de sensores wireless para monitoramento de temperatura, umidade, luz, pressão, etc** (MODUKURI et al., 2005) **Aplicação web - web services** (KIM; KIM, 2006; Ramakrishnan, 2009) **Sistema de controle de portão de estacionamento** (Ebnenasir, 2007) **Aplicação de combate a incêndios e resgate** (Hu; Indulska; Robinson, 2008) **Sistema multi-agentes** (CAPRARESCU, 2010) **Aplicação web - web services** (GORLA et al., 2010; den Hamer; Skramstad, 2011; HAOUAS; BOURCIER, 2012) **Sistema multicamadas autônomo** (Sliem; Salmi; Ioualalen, 2014) **Aplicação multi-robô** (Cui et al., 2014) **Sistema de transporte inteligente** (Cubo et al., 2015) **Smart Grids** (Lugo-Cordero; Guha; Ortiz-Rivera, 2014) **Robôs autônomos de limpeza** (GEROSTATHOPOULOS et al., 2019)
T4 - Retentar (2 estudos)	**Aplicação Web - Web services** (Ning-jiang Chen; Pan Lin, 2009) **Robôs autônomos** (STEINBAUER; MÖRTH; WOTAWA, 2006)
T5 - Híbridas (2 estudos)	**Sem domínio específico** (McGee; McGregor, 2016) Aplicação web - web services (KIM; KIM, 2006)
T6 - Outras (7 estudos)	**Aplicativo de computação pervasiva de escritório** (Cai; Peng; Zhang, 2012) **Sistema de controle de temperatura** (KLÖS; GÖTHEL; GLESNER, 2018) **Robôs autônomos** (XU; CHEUNG, 2005; HOFBAUR, 2007; Nair et al., 2019; GARVIN; COHEN; DWYER, 2013; CRESTANI; GODARY-DEJEAN; LAPIERRE, 2015)
T7 - Tratamento de exceções (7 estudos)	**Aplicativo de escritório baseado em agentes ciente de contexto** (DAMASCENO et al., 2006) **Rede de sensores wireless para monitoramento de incêndios, explosões e gases tóxicos** (Beder; Araujo, 2011) **Aplicação ciente de contexto de escritório** (Cho; Helal, 2011) **Sem domínio específico** (Rocha; Andrade, 2012; Yoon et al., 2014) **Robôs autônomos de limpeza** (Cho; Helal, 2012) **Sistema de controle de estacionamento** (Filho et al., 2014)

como técnicas de estimação de dados, baseadas em técnicas mais avançadas como as baseadas em Machine Learning, têm se destacado na avaliação de dados de sensores. Além da RSL para Cidades Inteligentes, foi conduzida uma RSL com foco em técnicas de tolerância a defeitos para Aplicações para Cidades Inteligentes. Tal RSL auxiliou no entendimento de técnicas e tipos de erros de aplicações, contribuindo para análise realizada na segunda RSL. Na próxima seção, é apresentada a proposta de trabalho apresentada nesta tese.

Tabela 18 – Tipos de defeitos específicos em SAs e SSCs.

ID	**Descrição**
DE-1	Um defeito em um reprodutor de áudio executando no dispositivo do usuário (KULKARNI; TRIPATHI, 2010)
DE-2	Um defeito em um sensor RFID (CHEN; YE; JACOBSEN, 2011)
DE-3	Um defeito em um sensor ultra-sônico (Xu et al., 2013)
DE-4	Um defeito em um sensor de batimentos cardíacos (ZAITER; HACINI, 2018)
DE-5	Um defeito na inicialização de uma classe factory (GORLA et al., 2010)
DE-6	Um defeito em um sensor que não é apto a enviar dados a outro sensor em uma rede de sensores wireless (MODUKURI et al., 2005)
DE-7	Um defeito em sensores de temperatura e fumaça (Hu; Indulska; Robinson, 2008)
DE-8	Um defeito em um sensor de frequência cardíaca (HAOUAS; BOURCIER, 2012)
DE-9	Um defeito em um recurso gerenciado (o recurso pode ser um servidor, um ciclo de instrução de CPU, espaço de memória, etc) (Sliem; Salmi; Ioualalen, 2014)
DE-10	Um defeito em um sensor de umidade (GEROSTATHOPOULOS et al., 2019)
DE-11	Um defeito em sensores de hardware ou em seus dispositivos de interface (câmeras e sensores ultrassônicos) (CRESTANI; GODARY-DEJEAN; LAPIERRE, 2015)
DE-12	Um defeito em atuadores ou em seus dispositivos de interface (CRESTANI; GODARY-DEJEAN; LAPIERRE, 2015)
DE-13	Um defeito em um atuador de um robô (Nair et al., 2019)
DE-14	Um defeito em um sensor de para-brisa (McGee; McGregor, 2016)

Capítulo 4

Uma Abordagem para Validação e Recuperação de Erros em Aplicações para Cidades Inteligentes

4.1 Considerações Iniciais

Neste capítulo é apresentada a proposta de trabalho desenvolvida nesta tese, na qual foram propostos:

- ❏ Uma abordagem para validação e recuperação de dados de sensores em Aplicações para Cidades Inteligentes como proposta principal desta tese.
- ❏ Uma proposta de middleware para atuar entre plataformas para Cidades Inteligentes e Aplicações para Cidades Inteligentes com o objetivo de implementar e consolidar a proposta de validação e recuperação.

Deste modo, neste capítulo, é apresentada a justificativa para a concepção da abordagem, a definição da abordagem de validação e recuperação de erros em dados de Aplicações para Cidades Inteligentes, a proposta de algoritmos para realizar a etapa de validação de dados e a proposta de um middleware como uma implementação da abordagem de validação.

4.2 Justificativa para Conceber a Abordagem

Sensores desempenham um papel fundamental na definição de estados de um sistema

(ALAG; AGOGINO; MORJARIA, 2001). É difícil garantir a confiabilidade de dados provenientes de sensores, de modo que, tais dados estejam atualizados e corretos (PIRES et al., 2016). Em cenários de erro, certos valores não são aceitáveis como corretos; por exemplo, um sensor de temperatura em uma sala de reuniões pode apresentar um valor máximo de 50 graus, mas em tal situação, a faixa de valores aceitáveis deveria estar entre 35 e 38 graus para aquele período. A partir de tal observação, alguns cenários podem ser descritos:

- **O valor do sensor está desatualizado** devido a algumas situações como, por exemplo, desconexão acidental dos sensores (uma vez que, eles estão espalhados pela cidade), consumo de bateria, superaquecimento, ou deterioração do material. Outro fator agravante são problemas de comunicação como, por exemplo, pacotes de dados perdidos ou entregues em atraso devido a problemas de comunicação entre os sensores e as aplicações. Por exemplo, um sensor de temperatura, durante o período da tarde marca uma temperatura de 18 graus enquanto deveria ser 35, talvez pelo fato da última leitura armazenada ter sido realizada durante o período matutino.

- **O valor do sensor está incorreto** em virtude de interferências durante o processo de leitura dos dados. Por exemplo, alguns sensores podem sofrer interferência eletromagnética, o que pode alterar o valor lido pelo sensor. Como consequência, o sensor pode apresentar valores fora dos valores adequados. Por exemplo, um sensor de temperatura, durante o período da tarde marca uma temperatura de 18 graus enquanto deveria ser 35, talvez pelo fato do sensor estar com defeito e realizando uma leitura incorreta mesmo que atualizada.

Sabe-se que Aplicações para Cidades Inteligentes são dependentes de contexto, e, além disso, dependentes de um contexto geral estabelecido por mais de uma variável de ambiente, ou seja, por mais de um contexto observado do ambiente. No entanto, um erro ativado por um defeito em pelo menos um dos contextos observados pode causar danos aos cidadãos e aos ambientes urbanos, dependendo da criticidade da aplicação. Na Tabela 19 são apresentados cenários, criados pela autora desta tese, nos quais sensores foram inseridos em um ambiente, exemplos de aplicação que utilizam tais sensores, impactos da aplicação (ou seja, do sistema em questão) na gestão da cidade, e impactos da ocorrência de falhas na aplicação do ponto de vista da gestão da cidade. Tais observações indicam que para uma cidade com um grande número de sensores implantados, a detecção de dados inválidos (ou seja, a validação de dados a fim de verificar se estão corretos) deve ser seriamente considerada como forma de contornar seus impactos nas aplicações e no ambiente urbano.

O processo de validação de dados de sensores é um processo importante em sistemas multi-sensores e consiste na verificação de condições dos dados e da validade dos mesmos a fim de obter dados acurados e confiáveis. A ideia consiste em validar dados

Cenários/Locais	**Aplicação**	**Impactos da aplicação na gestão da cidade**	**Impactos da falha da aplicação**
Ônibus, metrô e salas de metrôs	Aplicação que verifica a presença de fogo pela temperatura do ar	Garantia de segurança dos cidadãos dentro do metrô	Risco à vida dos cidadãos
Ônibus	Aplicação que controla a temperatura do ar condicionado do ônibus	Garantia de bem-estar dos cidadãos dentro do ônibus	Gasto de combustível em caso de uso desnecessário do ar-condicionado, e consequente emissão de poluentes
Parques	Aplicação que controla a irrigação de jardins de parques	Garantia de bem-estar dos cidadãos dentro do parque	Impacto financeiro, ou seja, a perda de investimento nos jardins, e desinteresse pelo parque por parte dos cidadãos
Avenidas	Aplicação que analisa a quantidade de carros que trafega em uma avenida	Garantia de bom funcionamento de semáforos inteligentes em situações de congestionamentos	Semáforos inteligentes podem operar de forma sub ótima, o que pode gerar congestionamentos ou atrasar situações de emergência

Tabela 19 – Cenários de uso de sensores e seu impacto em Aplicações para Cidades Inteligentes.

automaticamente em tempo de execução (PIRES et al., 2016). Com pouco ou nenhum poder de computação no sensor, é necessário aplicar mecanismos para detecção de defeitos (HENRY; WOOD, 2005). A relação entre confiabilidade, tolerância a defeitos e validação de dados de sensores pode ser estabelecida, uma vez que a segurança, confiabilidade e desempenho de aplicações multisensores são dependentes da confiabilidade e precisão de seus sensores (ALAG; AGOGINO; MORJARIA, 2001).

Com relação a tolerância a defeitos em Aplicações para Cidades Inteligentes, alguns estudos secundários indicam a falta de arquiteturas tolerantes a defeitos para tais aplicações. Por exemplo, Santana et al. (2017) conduziram uma RSL sobre plataformas, middlewares e arquiteturas para Aplicações para Cidades Inteligentes, elencando requisitos funcionais e não funcionais. Segundo Santana et al. (2017), os requisitos mais citados nos estudos foram interoperabilidade, segurança e escalabilidade. Além disso, os autores mencionam que um dos desafios dessas arquiteturas é garantir que todos os dados estejam corretos, uma vez que são provenientes de um ambiente com um grande número de fontes de dados. Os autores mencionam cinco plataformas/arquiteturas que propõem adaptar seu comportamento de acordo com o contexto para alcançar tolerância a defeitos (WAN et al., 2012; GIRTELSCHMID et al., 2013; GURGEN et al., 2013; PRIVAT; ZHAO; LEMKE, 2014; CHENG et al., 2015).

Em outro estudo secundário, realizado por Nascimento e Oliveira (2021), é apresentado um mapeamento sistemático da literatura a respeito de arquiteturas distribuídas de software sensível ao contexto para Cidades Inteligentes. O estudo fornece uma visão geral de tais arquiteturas e se concentra nos seguintes aspectos: descentralização, suporte a *plugins*, resiliência, fusão de dados, composição dinâmica, privacidade e segurança. De acordo com os autores, resiliência, em especial, é uma característica presente em poucos trabalhos analisados, sendo que, somente 8% (entre 72 estudos) apresentaram um mecanismo de tolerância a defeitos. O conceito de resiliência está relacionado a tolerância a

defeitos, uma vez que também inclui técnicas que são utilizadas para garantir a correta operação do sistema mesmo na presença de defeitos e ocorrência de falhas (PRADHAN et al., 2016). Assim, observa-se que a literatura especializada indica (1) a necessidade de esforços rumo ao desenvolvimento de arquiteturas tolerantes a defeitos para essas aplicações, assim como (2) a necessidade de garantir a entrega de dados confiáveis.

Conforme mencionado no Capítulo 2, a detecção de estados inconsistentes precede a recuperação de erros. Deste modo, desde que um dado é detectado como incorreto, deve-se aplicar um mecanismo de recuperação para completar as fases de tolerância a defeitos e garantir que o sistema retorne a um estado livre de erros. De acordo com os resultados obtidos na RSL sobre tolerância a defeitos em Aplicações para Cidades Inteligentes, a qual foi apresentada no Capítulo 3 desta tese, técnicas de estimação de dados baseadas em Machine Learning possuem características importantes para serem utilizadas na recuperação de dados (isto é, na recuperação de erros). Neste trabalho, propõe-se utilizar a vantagem do uso dessas técnicas tanto para validação quanto para recuperação de dados errôneos. Tais algoritmos possuem capacidade adaptativa, por utilizarem os próprios dados analisados para identificar padrões. Essa detecção de padrões pode servir tanto para detectar anomalias quando para estimar um valor adequado, isto é, um valor estimado automaticamente, com base nos próprios dados. Conforme descrito na Tabela 11, os estudos de Sasu, Puiu e Nechifor (2016), Puiu et al. (2016), Pastório e Camargo (2021), Tancev e Toro (2021) e (YUAN et al., 2021) propuseram técnicas de estimação de dados para recuperar dados de sensores de estados de erro.

Enquanto isso, apenas quatro estudos apresentaram estratégias de detecção de erros (ou seja, validação de dados) mais refinadas, que fogem à regra dos outros estudos que utilizam *timeouts*, ou seja, verificação de tempo de resposta, para ativar um mecanismo de tolerância a defeitos. O uso de *timeouts* auxilia no processo de garantir a entrega do dado, mas não garante que os dados sejam validados com relação à sua corretude (ou seja, precisão e validade) diante do contexto. Assim, observou-se que ainda há campo de exploração para estudos de algoritmos para validação de dados de sensores, como forma de compor uma abordagem de tolerância a defeitos mais completa. Deste modo, este trabalho apresenta e avalia o uso de algoritmos para validação de dados de sensores de Aplicações para Cidades Inteligentes. Na próxima seção será apresentada a proposta geral desta tese, na qual a validação de dados se insere.

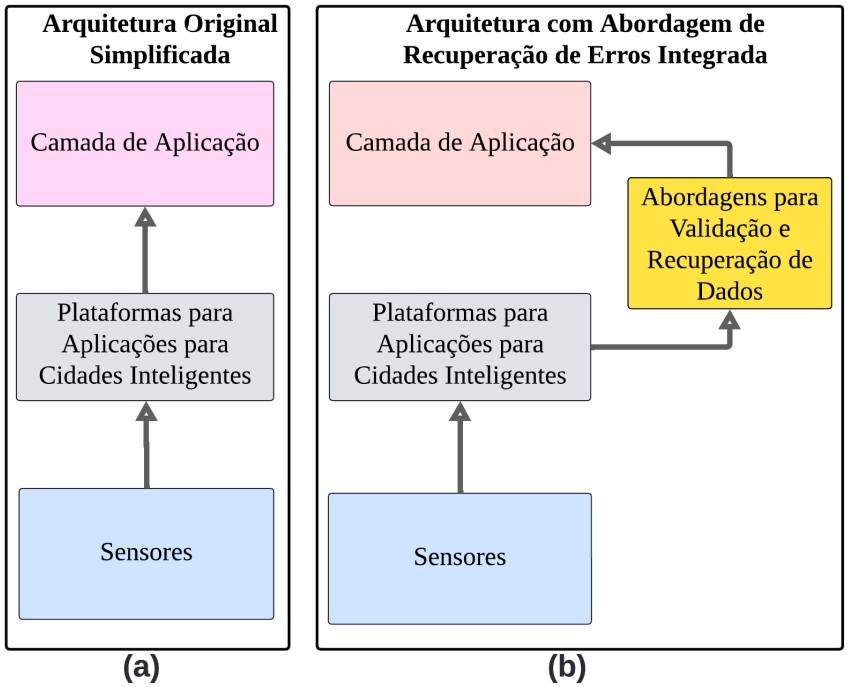
4.2.1 Definição da Abordagem

A abordagem descrita nesta seção é baseada em um cenário que considera a implantação de plataformas para Aplicações para Cidades Inteligentes. As plataformas de Aplicações para Cidades Inteligentes são fundamentais para melhorar o desenvolvimento, implantação, operação e integração de Aplicações para Cidades Inteligentes que suportam serviços direta ou indiretamente na cidade. Em geral, elas implementam um conjunto de

requisitos para fornecer serviços reutilizáveis, que permitem a integração de dispositivos de IoT, fornecendo recursos de armazenamento e processamento de dados (SANTANA et al., 2017). Tais plataformas suportam diferentes dispositivos físicos (ou seja, fontes de dados como, por exemplo, sensores) e também outros dispositivos de hardware com diferentes propósitos como, por exemplo, atuadores, dispositivos de rede, etc. Os dados disponíveis nessas plataformas se originam de diferentes recursos da cidade, ou seja, da infraestrutura da cidade, como ônibus, metrôs, parques, estacionamentos, etc. Esses dados são coletados por sensores espalhados na infraestrutura da cidade e, em seguida, armazenados em tais plataformas, que por sua vez fornecem ambientes para armazenamento e processamento de tais dados. O cenário original ao qual a abordagem aqui definida está inserida compreende uma camada de aplicações composta de aplicações externas com diferentes propósitos, que requisitam dados das plataformas para Aplicações para Cidades Inteligentes.

Na Figura 2 esboça-se o cenário da abordagem. Na Figura 2 (a) há o cenário original com plataformas e aplicações, enquanto na Figura 2 (b) há o cenário original com a inserção da proposta de validação e recuperação de dados de sensores entre as aplicações e as plataformas.

Figura 2 – Esquema Geral da Proposta. a) Cenário da proposta. b) Cenário da proposta com abordagem de validação e recuperação de erros.

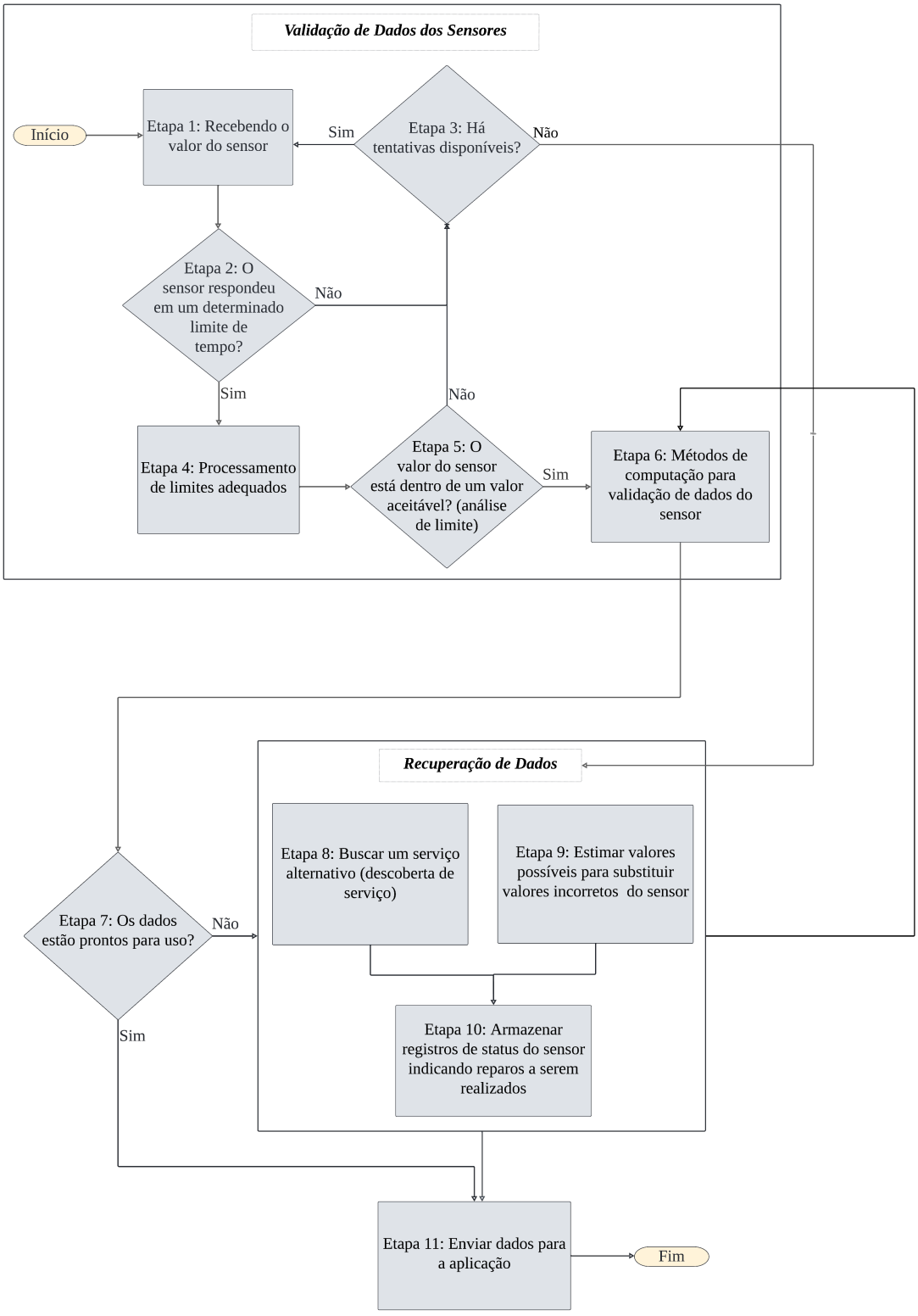


Fonte: Produzida pela autora.

Para desenvolver a abordagem apresentada na Figura 2, projetou-se um conjunto de etapas. Na Figura 3 exibe-se um fluxograma com tais etapas para realizar a validação

e recuperação de dados de sensores. As etapas enumeradas no fluxograma são descritas como:

Figura 3 – Fluxograma para as tarefas de validação e recuperação de dados de sensores.



Fonte: Produzida pela autora.

- *Etapa 1*: receber dados do sensor.
- *Etapa 2*: verificar o tempo de resposta do sensor; ou seja, verifica se o sensor respondeu em um limite de tempo.
- *Etapa 3*: verificar se existem tentativas disponíveis de conexão com o sensor; se sim, outra conexão é estabelecida e um novo dado é recebido do sensor.
- *Etapa 4*: criar e processar valores aceitáveis para o sensor dentro do contexto. Com esta verificação, pode-se assumir um erro de leitura, por exemplo. Para cada tipo de sensor, e de acordo com seu objetivo e especificação, tais faixas de valores podem ser determinadas.
- *Etapa 5*: esta etapa consiste em verificar se o valor está em um limite aceitável, de acordo com as faixas de valores pré-determinadas em Etapa 5.
- *Etapa 6*: utilizar métodos de validação de dados para verificar se os dados são válidos ou inválidos. Nesta etapa, métodos para a detecção de anomalias, ou seja, erros, podem ser utilizados a fim de identificar dados incorretos.
- *Etapa 7*: verificar o resultado obtido na etapa 6 para identificar se os dados estão prontos para serem utilizados pela aplicação; em outras palavras, esta etapa determina se os dados são válidos ou inválidos com base no resultado das etapas anteriores.
- *Etapa 8*: realizar a busca por uma fonte de dados alternativa; por exemplo, um sensor que esteja fisicamente próximo do sensor considerado defeituoso. Esta etapa será realizada quando o número de tentativas de conexão tiver sido excedido, ou quando os dados forem considerados inválidos.
- *Etapa 9*: calcular valores para substituir o valor inicial do sensor. Uma das soluções possíveis é calcular o valor do sensor com base no histórico, por exemplo, usando valores detectados ao longo do tempo para estimar o valor considerado inválido.
- *Etapa 10*: armazenar logs de status do sensor para gerar relatórios e, portanto, alertar sobre a necessidade de reparo manual ou mesmo o isolamento do sensor com defeito.
- *Etapa 11*: enviar dados validados (e possivelmente recuperados) para a aplicação.

4.2.2 Estudos da RSL sobre Técnicas para Validação de Dados de Sensores

A primeira grande etapa a ser realizada rumo a concretização da abordagem proposta é a validação de dados de sensores. Na RSL conduzida, quatro estudos se destacam por

utilizar técnicas mais refinadas para validar dados de sensores. Tais estudos são descritos a seguir.

Puiu et al. (2016) desenvolveram o framework CityPulse, que permite o desenvolvimento de aplicações que oferecem uma visão dinâmica e contínua da cidade. O CityPulse possui dois componentes de análise de dados: um componente de monitoramento da qualidade de dados, e um componente de recuperação de falhas. No componente de monitoramento da qualidade de dados são realizados dois tipos de monitoramento: o monitoramento atômico e monitoramento composto. O monitoramento atômico é a camada inferior do cálculo de qualidade e é responsável por calcular a qualidade dos dados (QoI - *Quality of Information* - Qualidade da Informação) em um nível individual, ou seja, para cada fluxo de dados provenientes de sensores. Para realizar o monitoramento atômico, é necessário especificar uma descrição do sensor, que define os parâmetros relevantes para a qualidade dos dados. Isso inclui informações como latência máxima, intervalo de atualização, e os tipos de dados esperados. Para isso, foram utilizados modelos de informação leves desenvolvidos sobre ontologias como *Semantic Sensor Network Ontology* e *Provenance Ontology*. Após receber as observações do fluxo de dados, o monitoramento atômico compara esses dados com a descrição do sensor e calcula a qualidade do fluxo com base em um algoritmo interno de classificação. O monitoramento composto, por sua vez, consiste em uma camada superior do cálculo de qualidade e combina informações de diferentes fluxos de dados para incluir várias observações adicionais no cálculo de qualidade. Assim, o monitoramento composto pode detectar fontes de informação defeituosas dentro de um grupo de fontes de dados, comparando-os entre si.

Yuan et al. (2021) apresentaram uma metodologia para fazer a avaliação de congestão de tráfego com foco em aplicações práticas em Xian, cidade da China. Foram utilizados dados de GPS de carros da cidade. O trabalho propôs um método para avaliar a possibilidade de congestionamentos. Além disso, o trabalho também teve como objetivo reduzir a dispersão dos dados utilizados e melhorar sua qualidade. Deste modo, as seguintes etapas foram realizadas: (i) Classificação de Dados Ruidosos: Foram estabelecidos critérios para longitude (< 107.4 ou > 109.49), latitude (< 33.42 ou > 34.45), velocidade (> 70 km/h) e direção ($> 359°$); esses valores são considerados fora do padrão esperado para os dados. (ii) Eliminação de Dados Ruidosos: foi realizada a filtragem dos dados, onde os registros identificados como ruidosos foram excluídos do conjunto para garantir que apenas os dados de alta qualidade fossem usados na análise. (iii) Redução Numérica dos Dados: embora o ruído tenha sido removido, tal redução é utilizada para reduzir o volume de dados, mantendo apenas informações essenciais para análise. (iv) Análise Estatística da Qualidade dos Dados: após a redução, foi realizada a medição da dispersão dos dados; a diminuição do coeficiente de dispersão indica dados mais concentrados em torno da média, sinalizando maior consistência e confiabilidade. (v) Melhoria na Precisão dos Dados: é utilizada a técnica de Map-Matching, ou seja, foi realizada uma correspondência

com outro mapa para corrigir imprecisões espaciais nos dados de GPS. Após essas etapas, dados foram simplificados e corrigidos, facilitando seu uso em análises mais complexas.

Khan, Kim e Park (2022) apresenta uma abordagem para detecção de anomalias em sequências de dados de sensores (dados de qualidade do ar) verificando valores menores ou iguais a zero e análise de mudanças rápidas nos valores por meio da observação de picos abruptos ou quedas inesperadas. Já o trabalho de Costa, Nassar e Dantas (2022) apresenta uma abordagem para detecção de anomalias. O trabalho apresenta a técnica Matriz de Perfil. Para a técnica de Matriz de Perfil, primeiro divide-se os dados em subsequências menores, isto é, em quartis. Em seguida, calcula-se uma distância (distância Euclidiana) entre cada quartil. Por fim é criado um perfil de distância que indica quais sequências podem ser de dados são normais. Após isso, uma análise automática utilizando limiares é utilizada para refinar a análise.

Comparando as abordagens utilizadas, os trabalhos apresentam técnicas baseadas em descrição de sensores (PUIU et al., 2016) e baseadas em análise de limites de valores pré-definidos (YUAN et al., 2021) requerem a intervenção de especialistas para a definição das descrições e dos limites de valores pré-definidos para diferentes contextos. Técnicas baseadas em análises de séries temporais também foram utilizadas. Khan, Kim e Park (2022) apresentaram uma abordagem para detecção de anomalias em sequências de dados de sensores, por meio da análise valores menores, iguais a zero, picos abruptos e diminuições rápidas nos dados. Contudo, tais avaliações podem dificultar a identificação de erros com mudanças sutis. O trabalho de Costa, Nassar e Dantas (2022) apresentou uma abordagem para detecção de anomalias por meio da técnica Matriz de Perfil. De acordo com os autores, esta abordagem se mostra promissora para cenários que exigem baixa latência.

Na Tabela 20 é apresentada a sumarização e contraste dos trabalhos relacionados. Dados os poucos estudos com foco em validação de dados e as limitações das abordagens observa-se que existe campo de exploração para o estudo de novos algoritmos para validação de dados, com o objetivo de maximizar a detecção de dados incorretos e diminuir tempo de processamento. A detecção de erros representados por mudanças sutis nos dados apresenta-se como um desafio, pois essas alterações podem estar dentro dos limites normais de variação, tornando-as difíceis de distinguir de flutuações naturais. Neste contexto, algoritmos baseados em séries temporais ganham destaque, pois permitem detectar padrões nos próprios dados aumentando a capacidade de generalização da solução para diferentes contextos e tipos de sensores. A seguir, são apresentados alguns algoritmos que podem ser utilizados para validar dados de sensores, a fim de compor a abordagem de validação de dados proposta.

Referência	Tipo de dado analisado	Técnica utilizada	Vantagens	Limitações
Puiu et al. (2016)	Dados de diferentes tipos de sensores	Descrição de sensores, ontologias e diversidade de fluxo de dados	Análise detalhada da qualidade dos dados	Depende da definição prévia das descrições dos sensores e da disponibilidade de diversos dispositivos fornecerem dados adequados
Yuan et al. (2021)	Dados de GPS	Filtros baseados em limites pré-definidos e Map-Matching	Melhoria na precisão dos dados	Pode excluir dados válidos se os limites não forem bem ajustados de acordo com o contexto
Khan et al. (2022)	Dados de qualidade do ar	Identificação de picos abruptos e quedas inesperadas nos dados	Simples de implementar e eficiente para mudanças rápidas	Apresenta dificuldade na detecção de erros sutis
Costa et al. (2022)	Dados do ambiente como temperatura e umidade	Matriz de Perfil	Baixa latência e análise automática	Pode ser sensível a variações menores

Tabela 20 – Comparação de abordagens de validação de dados de sensores.

4.3 Implementação da Etapa de Validação de Dados de Sensores

Neste trabalho, a etapa de validação de dados, isto é, a etapa 6 do fluxograma apresentado na Figura 3 foi implementada e avaliada. As causas que podem levar a dados incorretos são variadas, incluindo problemas mecânicos no sensor, falta de bateria, desgaste de componentes, falhas de comunicação, acidentes, interferências, e outros fatores já mencionados. Aplicar a validação em dados, que já passaram pelos processos de fusão (FADHEL et al., 2024) de dados[1] e armazenamento, pode ser vantajoso, pois erros provenientes de defeitos em qualquer um desses pontos de processamento podem ser mitigados.

Fadhel et al. (2024) mencionaram o poder transformador do aprendizado de máquina e do aprendizado profundo como responsável pela revolução das Cidades Inteligentes, pois ambos permitem reconhecer padrões, fazer previsões e automatizar tarefas. No estudo de Pires et al. (2016), os autores apresentaram um resumo de algoritmos e técnicas para validação e correção, ou seja, recuperação de dados de sensores. O estudo teve como foco aplicações móveis voltadas para a área da saúde, e apresenta algumas opções de algoritmos que podem ser explorados para validação de dados:

- **Redes Neurais Artificiais:** são modelos matemáticos inspirados no sistema nervoso central e podem ser treinadas para identificar sensores defeituosos e determinar um subconjunto ótimo de dados de sensores (BRANISAVLJEVIC; KAPELAN; PRODANOVIC, 2011; KASINATHAN et al., 2009).

- **Máquina de Vetores de Suporte** (do inglês, *Support Vector Machine– SVM*): é um método de aprendizado supervisionado que toma como entrada um conjunto de dados e prediz a qual. dentre duas classes, tal entrada pertence.

[1] A fusão de dados conecta fluxos de dados gerados por sensores, dispositivos e outros sistemas. Algoritmos de fusão de dados combinam dados para criar uma visão holística das condições urbanas.

- **Métodos Estatísticos:** envolvem o uso de média e desvio padrão que podem apontar a natureza incerta de sensores;
- **Distribuições Gaussianas:** tais distribuições são utilizadas para estimar se o valor decorrente de novos dados (por exemplo, dados oriundos de sensores) é suficientemente compatível com estimativas de um ponto estatístico específico.
- **Métodos de Redução de Dimensionalidade:** neste caso, a base central de todos esses métodos é que os dados obtidos de sensores são inicialmente comprimidos e posteriormente descompactados para o conjunto original de dados. O uso de Lógica Fuzzy também é uma opção para validar dados de sensores, e tempo envolve uma medida de desvio entre dois sinais e a aplicação de regras de inferência para determinar se existe ou não falha do sensor (HOLBERT; HEGER; ALANG-RASHID, 1994). Outra possibilidade é o uso da técnica Análise de Componentes Principais.
- **Filtros de Kalman:** é um método matemático que permite medir grandezas e gerar resultados que se aproximam de valores reais das grandezas medidas. Por exemplo, no trabalho de Shi (2012) os autores propuseram o uso de filtros de Kalman para a validação de dados de GPS.

4.3.1 Justificativa sobre a Escolha dos Algoritmos para Validação de Dados de Sensores

Algoritmos de aprendizagem não-supervisionada são uma boa escolha para dados de Cidades Inteligentes, pois a rotulagem de dados é impraticável considerando ambientes dinâmicos como, por exemplo, o monitoramento de sensores em tempo real. Dadas as características desses algoritmos e as limitações das abordagens de validação presentes na literatura, dois algoritmos de aprendizagem não-supervisionada se destacam na detecção de anomalias, sendo eles, Isolation Forest e Support Vector Machines (SVM) One-Class. O primeiro se destaca por sua característica de tempo de processamento de detecção (FARIZI; HIDAYAH; RIZAL, 2021), enquanto o segundo se destaca por ser uma abordagem tradicional e sólida para tarefas de detecção de anomalias, sendo uma extensão do método SVM mencionado por Pires et al. (2016). Para compará-los com uma abordagem tradicional no contexto de tolerância a defeitos, foi avaliada também uma técnica de Diversidade por correlação. A seguir, apresenta-se uma explicação conceitual breve desses três algoritmos.

Ressalta-se que a detecção de anomalias, ou detecção de *outliers*, permite determinar quais instâncias se destacam como sendo anormais em um conjunto de dados. Uma anomalia pode ser definida como uma observação que desvia demasiadamente das outras, gerando dúvidas a respeito de sua fonte (AHMED; Naser Mahmood; HU, 2016). As anomalias podem estar relacionadas a erros, indicando que alguma ação pode ser tomada

para corrigi-las. A detecção de anomalias tem sido utilizada em vários domínios de aplicação, tais como, detecção de fraudes, detecção de intrusão, processamento de imagens, e instabilidade em redes (AHMED; MAHMOOD; ISLAM, 2016). Ambos os métodos de aprendizado não-supervisionado avaliados nesta tese, sendo eles Isolation Forest e SVM, são algoritmos de detecção de anomalias.

O terceiro algoritmo avaliado é baseado na técnica de Diversidade, bem estabelecida na comunidade de tolerância a defeitos. O objetivo foi contrastar técnicas de detecção de anomalias com uma estratégia bem conhecida na comunidade de tolerância a defeitos. A seguir é apresentada uma descrição dos três algoritmos.

4.3.1.1 Isolation Forest

Isolation Forest (LIU; TING; ZHOU, 2008) é um método de detecção de anomalias inspirado no algoritmo de Random Forest (V et al., 2003). Isolation Forest tem se destacado com relação a desempenho em diversas aplicações na detecção de anomalias (ZHANG et al., 2011). Apesar de sua simplicidade, tal algoritmo se destaca por sua capacidade de lidar com dados de alta dimensão e por sua velocidade (FARIZI; HIDAYAH; RIZAL, 2021), o que é uma característica importante no cenário de Aplicações para Cidades Inteligentes.

A premissa básica do algoritmo é que as anomalias são poucas e destoam muito dos demais elementos e, então, podem ser separados pelo processo de isolamento, que ocorre a partir da geração de uma árvore. Em uma árvore aleatória induzida por dados, o particionamento de instâncias é repetido recursivamente até que todas as instâncias sejam isoladas. O processo de isolamento de elementos é repetido iterativamente, gerando diferentes configurações de árvores. Após isso, os outliers são identificados, tomando-se como base o custo para que determinado elemento seja isolado dos demais, basicamente observando-se a distância entre a raiz da árvore para cada nova observação. O comprimento esperado do caminho h(x) para isolar uma instância x em uma árvore é uma medida direta do número de divisões necessárias para separá-la (LIU; TING; ZHOU, 2008).

A pontuação de anomalia é dada por *s(x, n)*, que é a pontuação de isolamento para o ponto de dados x mediante um número de instâncias. A pontuação é calculada de acordo com a Equação 1:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{1}$$

Em que:

- $E(h(x))$: É o comprimento médio do caminho de x em todas as árvores.
- $c(n)$: É o valor esperado do comprimento do caminho para uma árvore binária com n instâncias, dado pela Equação 2 :

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n} \tag{2}$$

- $H(i)$: É a i-ésima constante harmônica, definida pela Equação 3:

$$H(i) = \sum_{k=1}^{i} \frac{1}{k} \tag{3}$$

Durante o processamento do algoritmo, um escore (ou seja, uma pontuação de anomalias) é criado. Se o escore retornar um valor muito próximo de 1, então o ponto analisado é definitivamente uma anomalia. Se o escore apresentar um valor muito menor que 0.5, as instâncias são consideradas instâncias normais. Se o escore retornar valores próximos de 0.5, não há padrão claro de anomalia. Na implementação do Isolation Forest avaliada nesta tese, especificamente usando-se a biblioteca *scikit-learn* da linguagem Python, o valor -1 para anomalias é uma convenção utilizada para facilitar a interpretação dos resultados, o que é útil em pipelines nos quais outras técnicas também utilizam essa convenção.

Dada a formulação do algoritmo, os principais parâmetros sujeitos a ajustes do Isolation Forest são:

- Número de estimadores *(n-estimators)*: É o número de árvores de decisão, que afeta a profundidade das árvores.
- Número de amostras *(max-samples)*: É o número de amostras por árvore.
- Número de *outliers (contamination)*: É a proporção estimada de *outliers* no conjunto de dados.
- Número de características *(max-features)*: É o percentual de características (ou atributos) a serem usadas para construir cada árvore.
- Aleatoriedade *(random-state)*: Define a aleatoriedade usada no treinamento do modelo e afeta a forma como as amostras e as divisões são feitas nas árvores.

4.3.1.2 Suport Vector Machine (SVM) One-Class

O Support Vector Machine (SVM) One-Class (SCHöLKOPF et al., 2001) é uma variante do SVM (CORTES; VAPNIK, 1995), sendo um algoritmo utilizado para detecção de anomalias. O SVM One-Class caracteriza-se também como um algoritmo de aprendizado não-supervisionado, no qual os dados não possuem rótulos, ou seja, as amostras não possuem categorias pré-definidas. Então, o treinamento consiste em ajustar o modelo com base nos próprios dados fornecidos e não rotulados, a fim de identificar padrões.

O SVM One-Class computa uma função binária que captura regiões no espaço de entrada onde a densidade de probabilidade se localiza, ou seja, onde a maioria dos dados

reside e, assim sendo, onde a função tem valor diferente de zero. O objetivo é encontrar uma função $f(x)$ tal que $f(x) > 0$ para os pontos normais (dentro do suporte da distribuição) e $f(x) < 0$ para os *outliers*, isto é, dados anômalos.

Para separar o conjunto de dados da origem, deve-se resolver a seguinte função objetivo, conforme Equação 4:

$$\min_{\mathbf{w},\rho,\xi} \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \tag{4}$$

Sujeito às restrições da Equação 5:

$$(\mathbf{w} \cdot \phi(\mathbf{x}_i)) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{5}$$

Em que:

- $\mathbf{w}$: É o vetor de pesos que define o hiperplano.
- $\phi(\mathbf{x}_i)$: É a função de transformação (kernel) que mapeia os dados para um espaço de alta dimensão.
- ρ: Offset (margem).
- ξ_i: São variáveis de folga (*slack*), que permitem que alguns pontos sejam considerados outliers.
- $\nu \in (0, 1]$: É o parâmetro que controla a fração de pontos esperados como outliers e o número de vetores de suporte.
- n: É o número de amostras.

Considerando as restrições, tem-se que dados normais são pontos que satisfazem:

$$\mathbf{w} \cdot \phi(\mathbf{x}_i) \geq \rho$$

.

Já os *outliers* são pontos para os quais:

$$\mathbf{w} \cdot \phi(\mathbf{x}_i) < \rho,$$

indicando que os pontos estão fora da margem definida e são considerados anômalos. A tolerância é controlada por ξ_i, que são variáveis que permitem que alguns pontos normais sejam tratados como *outliers* devido a limitações de ajuste do modelo.

Dada a formulação do algoritmo, os principais parâmetros sujeitos a ajustes do SVM One-Class são:

- γ (gamma): Controla o alcance da função kernel $\phi(\mathbf{x}_i)$ da Equação 5.
- ν (nu): Controla a fração de pontos que o modelo classifica como *outliers*.

4.3.1.3 Diversidade Baseada em Correlação

O algoritmo de Diversidade por Correlação é uma abordagem empregada neste trabalho para a detecção de anomalias que explora a ideia de combinar dados de sensores com dados semelhantes a fim de comparar os valores e identificar erros. O algoritmo consiste em: (i) identificar um sensor equivalente, ou seja, com valores de leituras bastante próximos, isto é, de um sensor que faça uma leitura de mesmo tipo e de mesma medida; e (ii) verificar se o valor que se deseja validar é próximo do valor do sensor análogo. Para identificar os sensores análogos, isto é, identificar sensores de leituras parecidas, foi utilizado o coeficiente de correlação de Pearson. Tal coeficiente é muito utilizado e consolidado para identificação de similaridades, descrito conforme Equação 6:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6}$$

Em que:

- x_i e y_i são os valores das variáveis X e Y, respectivamente, neste trabalho representando os valores de pares dos sensores.
- $\bar{x}$ e $\bar{y}$ são as médias das variáveis X e Y.
- n é o número de pares de dados.

Um valor de coeficiente próximo de 1 indica correlação forte e próximo de 0 correlação fraca.

A seguir é apresentada a proposta de um middleware para consolidar a etapa de validação de dados, mais especificamente a etapa 6, apresentada no fluxograma da Figura 3, para o cenário de Aplicações para Cidades Inteligentes.

4.3.2 Um Middleware como uma Implementação da Abordagem de Validação de Dados de Sensores de Aplicações para Cidades Inteligentes

Um middleware pode ser definido como um componente que fornece um conjunto de abstrações para programação, assim facilitando a integração e comunicação de componentes heterogêneos (FERSI, 2015). Por exemplo, um middleware de IoT atua como um vínculo que une diferentes componentes de algum aplicativo, o quais se comunicam por meio de uma rede heterogênea. Um middleware também fornece uma API (Application Programming Interface) para estabelecer a comunicação entre diferentes camadas e serviços requeridos por aplicativos (BANDYOPADHYAY et al., 2011). De acordo com (CHELLOUG; EL-ZAWAWY, 2017), alguns estudos fornecem evidências de que a Arquitetura Orientada a Serviços (SOA) é uma maneira eficiente de construir middlewares que

lidam com uma grande variedade de dados, como dados que vêm de vários dispositivos de IoT. Este tipo de arquitetura é usado para gerenciar serviços, por meio da incorporação de um provedor de serviços, para hospedar um ou mais serviços, um consumidor de serviços, e um registro de serviços (CHELLOUG; EL-ZAWAWY, 2017).

O middleware desenvolvido, moldado como um serviço e ainda em uma versão preliminar, provê uma solução referente à abordagem de validação de dados descrita neste capítulo. A solução, quando completamente implementada, irá desempenhar as funções de validação e recuperação de dados. A sua proposta de utilização é para diferentes tipos de sensores, podendo ser utilizado para diferentes tipos de aplicações de Cidades Inteligentes. Assim, foram projetados os seguintes requisitos funcionais para o middleware:

1. ***Requisito Funcional de Validação de Dados***: o sistema deve detectar valores inválidos obtidos de sensores após sua aquisição e armazenamento.

2. ***Requisito Funcional de Recuperação de Dados***: o sistema deve ser capaz de recuperar dados, ou seja, obter o valores contextuais, ou estimar os valores contextuais, de dados avaliados como incorretos.

Para implementar os requisitos mencionados acima, propõe-se um componente que engloba as tarefas de validar e recuperar dados. O componente mencionado é apresentado na Figura 4 com o título de *Componente de Recuperação de Erros*. Na figura é apresentada a relação entre o componente proposto e a plataforma baseada em microsserviços InterSCity (ESPOSTE et al., 2017a), selecionada como plataforma inicial para a realização de testes da abordagem de validação proposta neste trabalho. Santana et al. (2017) revisaram um conjunto de diferentes plataformas para Cidades Inteligentes apresentadas pela literatura científica. A plataforma InterSCity foi escolhida por ser uma plataforma acessível para implantação e teste em máquinas locais, e por possuir capacidade de integração com diferentes aplicações por seguir uma arquitetura baseada em microserviços. Além disso, a plataforma possui documentação detalhada e comunidade ativa para apoio a dúvidas.

A plataforma InterSCity segue uma arquitetura baseada em microserviços e possui uma infraestrutura tecnológica unificada para serviços em larga escala. A plataforma conecta diversos dispositivos físicos e serviços de diferentes tipos. Os microsserviços têm a capacidade de se comunicar por meio de suas APIs REST (Representational State Transfer) ou de maneira assíncrona usando o RabbitMQ como barramento de mensagens e aplicando o padrão de design *publish-subscribe*. No modelo *publish-subscribe*, um broker distribui uma mensagem para seus assinantes. O RabbitMQ é uma plataforma de mensageria que permite que sistemas distribuídos troquem informações de maneira assíncrona, ou seja, a comunicação ou operação é não-bloqueante, na qual o remetente faz a requisição e não espera por uma resposta imediata. Informações sobre tecnologias utilizadas na plataforma estão disponíveis em sua documentação (ESPOSTE et al., 2017b).

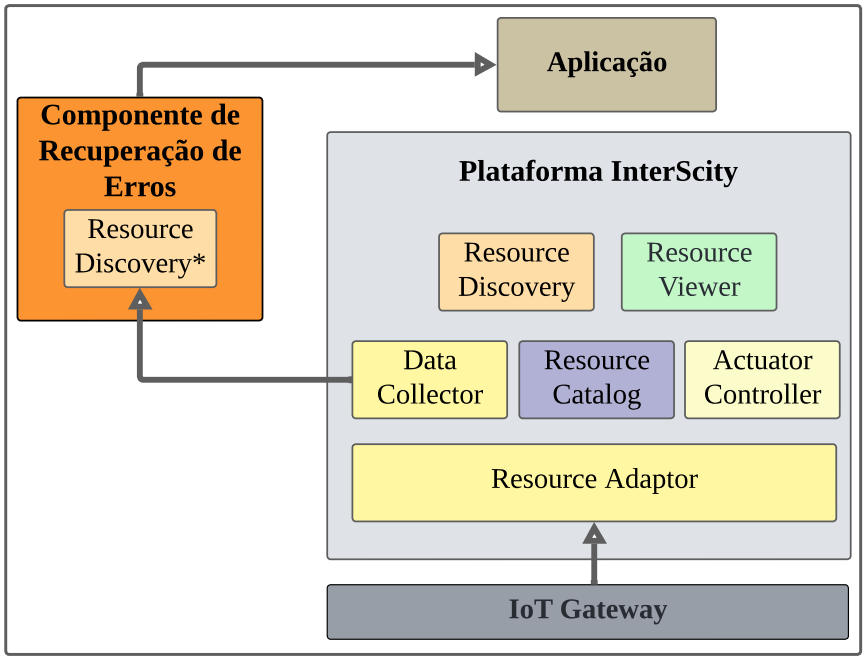
Os microserviços que compõem a plataforma InterScity são os seguintes: Resource Adaptor para integração dos dispositivos de IoT, Resource Catalog, Data Collector e Actuator Controller para o gerenciamento de dados, Resource Discovery para a descoberta de recursos e o Resource Viewer para visualização (ESPOSTE et al., 2017a).

Os IoT Gateways, conforme exibidos na Figura 4, servem para adicionar novos serviços na plataforma. Uma API Gateway é um elemento de software que funciona como um ponto de acesso unificado, responsável por gerenciar e redirecionar as requisições realizadas por clientes a vários serviços distintos. Na plataforma InterSCity, um IoT Gateway publica novos dados de sensores para a plataforma usando a API REST do Resource Adaptor. Este último facilita a comunicação entre serviços externos e a plataforma, permite enviar dados coletados por esses recursos (por exemplo, sensores) para a plataforma e permite assinar eventos para receber notificações sobre comandos que precisam ser enviados para atuadores. O Resource Adaptor registra novos dados de recursos da plataforma, posta e atualiza dados dos mesmos. O Resource Catalog gera identificadores únicos para os recursos cadastrados que podem ser sensores ou atuadores (ESPOSTE et al., 2017a).

O microserviço chamado Data Collector armazena informações de sensores. As informações de um sensor refere-se a dados contextuais ocorridos em determinado tempo. Tal serviço oferece uma API para acesso às informações sobre os recursos da cidade, incluindo dados históricos. O microserviço Actuator Controller, por outro lado, realiza os pedidos de atuação sobre os recursos da cidade que possuem a função de atuar, mantendo históricos de atuação. O microserviço Resource Discoverer fornece uma API para descobrir recursos disponíveis na cidade, permitindo a configuração de filtros. Resource Viewer é um microsserviço destinado à exibição de informações sobre os recursos de forma visual, usando os microserviços Resource Catalog e Data Collector (ESPOSTE et al., 2017a).

Na Figura 4 apresenta-se o fluxo de dados entre uma aplicação externa a InterSCity. Essa aplicação, que pode ser de diferentes características, consome dados, – isto é, recursos – que foram cadastrados na plataforma. Na figura, é apresentado o esquema da proposta, em que os dados, antes de serem enviados para a aplicação, passam pelas etapas de validação e recuperação de dados. Como consequência, espera-se que dados mais confiáveis podem ser consumidos pela aplicação. A proposta do middleware é implementar as abordagens de validação e recuperação elaboradas para que sejam utilizados pelas plataformas. A ideia do componente Resource Discovery* envolve realizar a descoberta de dados. Por exemplo, no caso da InterSCity, o componente permite realizar a busca pelo recurso, e seus dados, por meio de um identificador, por data, por local, etc. Assim, o componente Resource Discovery* permitiria a adaptação do uso do middleware para diferentes plataformas por meio do uso de APIs. Do ponto de vista de um engenheiro de aplicações, o middleware irá servir como uma camada de software intermediária.

Figura 4 – Arquitetura da plataforma InterSCity com a acoplamento do middleware de validação e recuperação proposto. A seta representa o fluxo de dados entre a plataforma, o middleware e a aplicação.



Fonte: Produzida pela autora.

4.4 Considerações Finais

Neste capítulo, primeiramente, delinearam-se cenários de erros em Aplicações para Cidades Inteligentes. Em seguida, outros estudos secundários relacionados à RSL apresentada no Capítulo 3) foram revisitados com a intenção de ressaltar a importância de técnicas de tolerância a defeitos em Aplicações para Cidades Inteligentes. Depois, estudos da RSL foram revisitados a fim de justificar a utilização dos algoritmos selecionados para validação de dados. Por fim, apresentou-se uma proposta de middleware como implementação da solução de validação de dados no contexto de Cidades Inteligentes.

Dadas as estratégias de validação de dados propostas nesta tese, foi desenvolvido um protocolo experimental a fim de avaliar os algoritmos de validação selecionados no contexto de dados de Aplicações para Cidades Inteligentes. A seguir, é apresentado o protocolo experimental e a abordagem experimental desenvolvida para avaliar os algoritmos de validação.

Capítulo 5

Uma Abordagem Experimental para Validação de Dados de Sensores de Aplicações para Cidades Inteligentes

5.1 Considerações Iniciais

Neste capítulo são descritos os experimentos realizados para avaliar os algoritmos de detecção de anomalias, ou seja, os algoritmos selecionados de detecção de erros. Para isso, foi criado um protocolo experimental com base em um cenário de Aplicações para Cidades Inteligentes e em diferentes cenários de erro. Os resultados dos experimentos foram avaliados para um conjunto de sensores com base em duas métricas de avaliação. Em seguida, testes estatísticos foram realizados a fim de analisar a relevância dos resultados. Sendo assim, a seguir é apresentado o planejamento do experimento (Seção 5.2), os resultados obtidos (Seções 5.3), análises comparativas diretas dos resultados e testes estatísticos apresentados (Seção 5.4), e ao final, respostas às hipóteses e as conclusões obtidas (Seção 5.5). Por fim, são apresentadas as ameaças à validade do experimento (Seção 5.6).

5.2 Planejamento do Experimento

5.2.1 Contexto Experimental

O contexto deste experimento envolve dados de sensores de Aplicações para Cidades Inteligentes que estão espalhados pela cidade e são utilizados para realizar uma leitura

de contexto, ou seja, para monitorar variáveis do ambiente urbano. O objetivo geral do experimento é a detecção automática de estados de erro em dados provenientes de sensores com o objetivo de validá-los e torná-los disponíveis e adequados para as aplicações. A seguir, é apresentado o planejamento do experimento realizado de acordo com as principais diretrizes sobre experimentação no campo da Engenharia de Software (WOHLIN et al., 2012).

5.2.2 Identificação e Objetivo

Título: Avaliação Comparativa de Algoritmos de Detecção de Anomalias para Validação de Dados de Sensores em Aplicações para Cidades Inteligentes.

Área técnica: Engenharia de Software

Autor: Kathiani Elisa de Souza

Afiliação: Universidade Federal de São Carlos

Local: Departamento de Computação, São Carlos, São Paulo, Brasil

Contexto: Cidades inteligentes

De acordo com o modelo proposto por Basili (1992), o objetivo específico deste experimento é:

- ❏ Analisar *algoritmos de detecção de anomalias*,
- ❏ Com a finalidade de compará-los,
- ❏ Com respeito à *medida de detecção F1-Measure e tempo de computação*,
- ❏ Do ponto de vista de pesquisador,
- ❏ No contexto de dados simulados de sensores de Aplicações para Cidades Inteligentes.

5.2.3 Objeto Experimental

Neste experimento, três algoritmos foram avaliados, a saber: (i) Diversidade por Correlação, (ii) Isolation Forest, e (iii) Suport Vector Machine (SVM) One-Class. Além disso, dados simulados de Aplicações para Cidades Inteligentes foram manipulados e analisados para testar tais algoritmos.

5.2.4 Seleção das Variáveis

As *Variáveis Independentes* do experimento são:

1. Dois algoritmos de detecção de anomalias, a saber, Isolation Forest e SVM One-Class, e um algoritmo baseado na técnica de Diversidade.
2. Quatro tipos de erro injetados na sequência de dados, a saber: Bias, Freezing, Loss of Accuracy e Noise.
3. A porcentagem de dados incorretos na sequência de dados de 1 dia de leitura, a saber: 10 por cento e 20 por cento.

As *Variáveis Dependentes* deste experimento são:

1. A taxa de detecção de erros, em termos da medida F1-Measure calculada para cada sensor.
2. O tempo de processamento de cada algoritmo (em segundos) para cada sensor.

5.2.5 Formulação das Hipóteses

Neste trabalho, quatro hipótese nulas foram avaliadas conforme descrito a seguir.

5.2.5.1 Hipóteses para Detecção de Erros

- **Hipótese Nula (H1-0)**: Não há diferença significativa na medida F1-Measure na detecção de erros entre o algoritmo SVM One-Class e o algoritmo Isolation Forest, na detecção de todos os tipos de erros, com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
  - **Hipótese Alternativa (H1-1)**: Há diferença significativa na medida F1-Measure na detecção de erros entre o algoritmo SVM One-Class e o algoritmo Isolation Forest, na detecção de todos os tipos de erros, com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
- **Hipótese Nula (H2-0):** Não há diferença significativa na medida F1-Measure na detecção de erros entre a técnica Diversidade e o melhor método de detecção de anomalias (SVM One-Class ou Isolation Forest), na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
  - **Hipótese Alternativa (H2-1):** Há diferença significativa na medida F1-Measure na detecção de erros entre a técnia Diversidade e o melhor método

de detecção de anomalias (SVM One-Class ou Isolation Forest), na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).

5.2.5.2 Hipóteses para Tempo de Computação

- **Hipótese Nula (H3-0):** Não há diferença significativa no tempo médio de processamento entre o algoritmo SVM One-Class e o algoritmo Isolation Forest na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
  - **Hipótese Alternativa (H3-1):** Há diferença significativa no tempo médio de processamento entre o algoritmo SVM One-Class e o algoritmo Isolation Forest na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
- **Hipótese Nula (H4-0):** Não há diferença significativa no tempo médio de processamento entre o algoritmo Diversidade e o melhor método de detecção de anomalias (SVM One-Class ou Isolation Forest) na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).
  - **Hipótese Alternativa (H4-1):** Não há diferença significativa no tempo médio de processamento entre o algoritmo Diversidade e o melhor método de detecção de anomalias (SVM One-Class ou Isolation Forest) na detecção de todos os tipos de erros com diferentes porcentagens de erro inseridas (para 10% e 20% de erros injetados).

5.2.6 Seleção dos Sujeitos

Como sujeitos do experimento definiu-se um número 30 de sensores cujos dados foram simulados e analisados posteriormente.

5.2.7 Amostragem e Aleatorização

O objetivo da amostragem foi utilizar amostras representativas de dados típicos de aplicações para cidades inteligentes. Para isso, foram simuladas sequências de dados para o cenário de sensores de temperatura. Os experimentos foram executados com dois lotes de dados distintos de acordo com a quantidade de erros injetados. Os algoritmos foram executados em uma ordem pré-determinada, o que pode beneficiar um algoritmo por efeitos de aquecimento do processador. Em adição, cada algoritmo foi exposto aos mesmos lotes de dados sob as mesmas condições experimentais.

5.2.7.1 Dados de Sensores de Temperatura

Foram geradas sequências não-sazonais, isto é, séries temporais que apresentam padrões que se repetem. As sequências não-sazonais, para os quais a análise do experimento foi realizada, foram geradas conforme descrito abaixo:

- ❑ Foram criados dados de 24 horas de leitura realizada pelos sensores de temperatura, contabilizando 8.640 leituras (a cada 10 segundos), com pequenas flutuações aleatórias ao redor de 42 graus, dado que no contexto de uma grande cidade as sensores podem apresentar pequenas flutuações.
- ❑ Foram gerados dois lotes de dados distintos, cada lote contendo dados de 30 sensores.

Os dados utilizados foram simulados, ou seja, criados automaticamente, pois a simulação permite gerar grandes volumes de dados rapidamente.

5.2.8 Etapas do Experimento

Este experimento, desde a simulação de dados de sensores até a avaliação dos algoritmos de validação, foi elaborado nas quatro etapas que são descritas a seguir.

5.2.9 Etapa 1: Simular Dados de Sensores

Nesta etapa, foi realizada a simulação de dados, na qual dados foram gerados para teste, conforme descrito na Seção 5.2.7.

5.2.10 Etapa 2: Injetar Erros nos Dados Simulados

Injeção de falhas e erros são assuntos de pesquisa populares nas comunidade científica de confiabilidade e dependabilidade de software. Técnicas de injeção de erros são usadas para avaliar a técnicas de tolerância a defeitos e também a confiabilidade do sistema. Tais técnicas permitem avaliar o comportamento do sistema na presença de erros e determinar a cobertura de abordagens de detecção e recuperação de erros (Kooli; Di Natale, 2014). Nesta etapa, foram inseridos diferentes tipos de erros nas sequências de dados simuladas. Os erros inseridos tratam-se de erros episódicos, ou seja, não acontecem repetidamente na sequência e não possuem um padrão de local na sequência.

Para realizar o teste dos algoritmos os seguintes tipos de erros foram injetados:

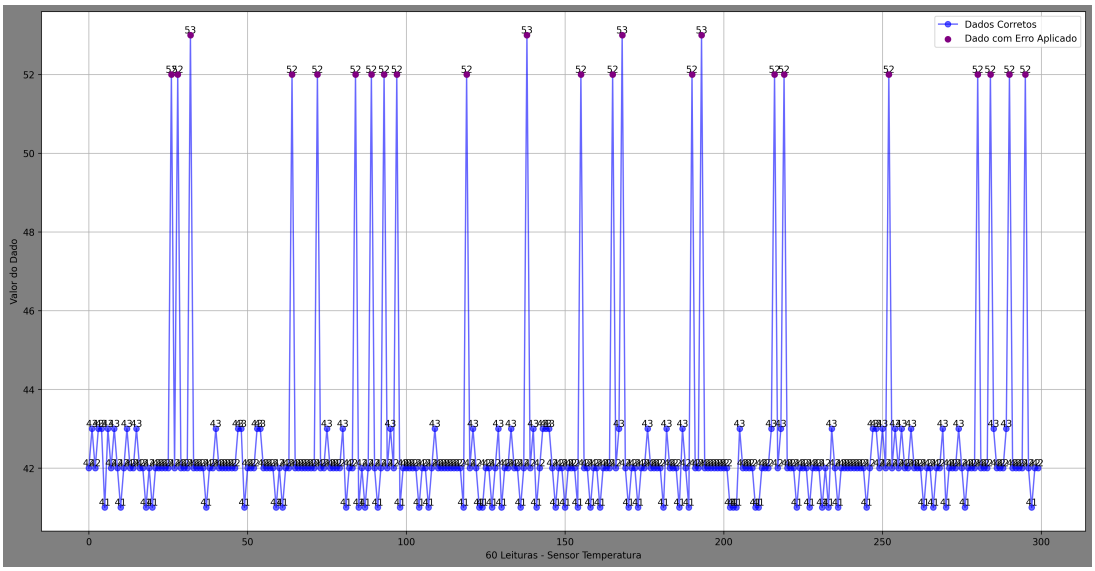
- ❑ *Bias*: a saída do sensor tem uma diferença (constante) do valor correto.
- ❑ *Noise*: a saída aumenta ou diminui, com intensidade variável nos incrementos ou decrementos no valor de leitura do sensor.
- ❑ *Loss of Accuracy*: a saída aumenta ou diminui suavemente a partir do valor correto.

- *Freezing / Stuck-at*: a saída fica presa em um valor fixo a partir de determinado valor correto.

Os quatro tipos de erros foram aplicados nos dois lotes de dados (contendo 30 sensores cada). No caso do Freezing, ambos foram paralisados a partir do meio da sequência. Os erros de Bias, Noise, Loss of Accuracy e Freezing são apresentados nas Figuras 5, 6, 7 e 8, respectivamente. Em todas as figuras foram plotados os erros injetados nos 300 primeiros dados da sequência para o sensor 1 para o lote 1, exceto para a figura relacionada ao erro Freezing, na qual foi efetuado um recorte nos dados a partir do centro.

Os erros foram simulados, pois permitem testar cenários de erro específicos que podem ser difíceis de observar em dados reais.

Figura 5 – Erro de Bias.



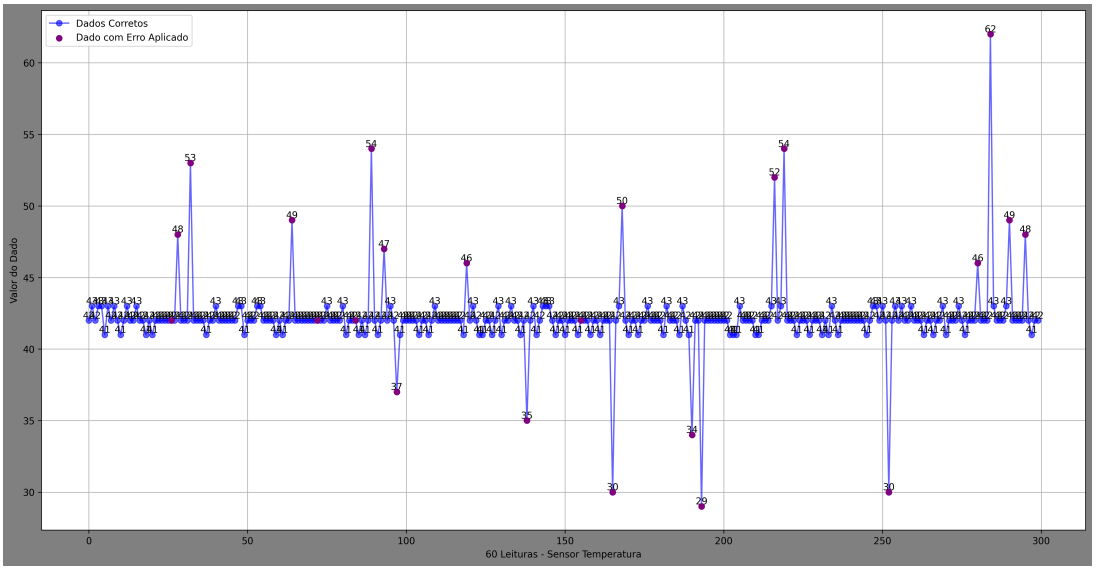
Fonte: Produzida pela autora.

5.2.11 Etapa 3: Criar Grupos Experimentais

Grupos experimentais foram criados a fim de avaliar os algoritmos de detecção de anomalias sob diferentes situações. Para cada grupo experimental foram realizadas 8.640 análises, provenientes da coleta em um período de 24 horas com leituras de 10 em 10 segundos. Os grupos de execução foram configurados para um dado sensor Y representando um sensor de temperatura. Os grupos de execução foram definidos de acordo com a Equação 7:

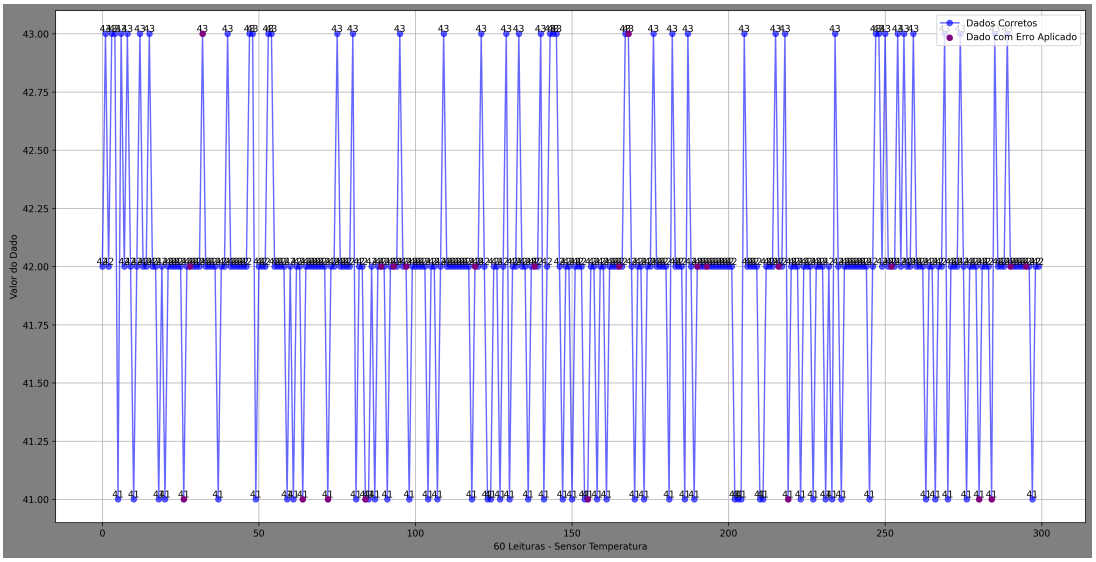
$$G_{z,y} = X \times K \times N \tag{7}$$

Figura 6 – Erro de Noise.



Fonte: Produzida pela autora.

Figura 7 – Erro de Loss of Accuracy.



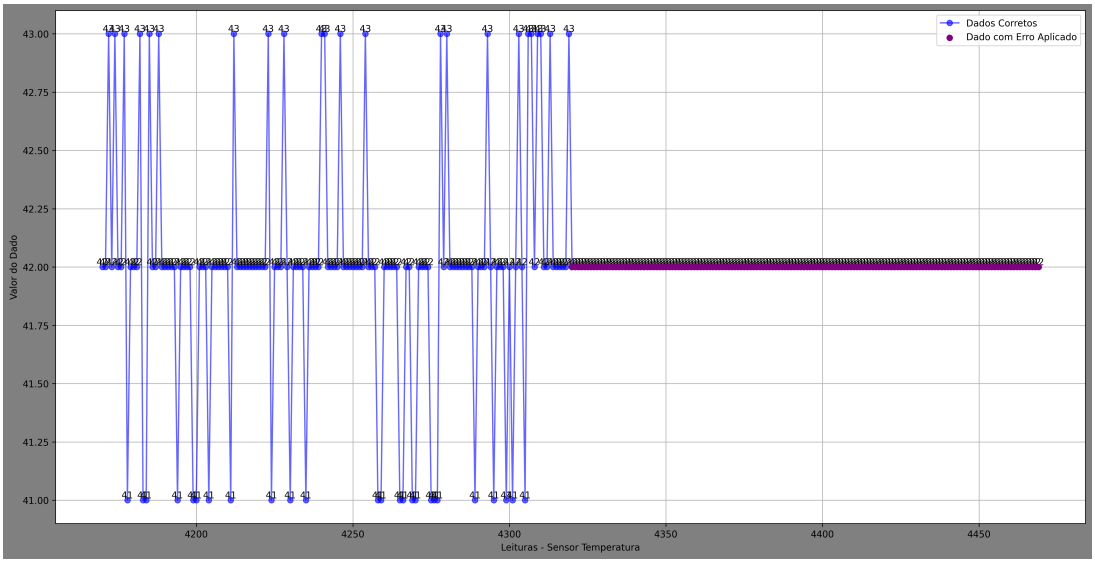
Fonte: Produzida pela autora.

Na Equação 7, z representa o número identificador do grupo experimental e y identificador do sensor para o qual o grupo foi realizado. Os demais itens da equação foram configurados da seguinte maneira:

- $X = \{x1, x2, x3\}$, onde x representa os algoritmos testados: x1 = Diversidade por Correlação, x2 = Isolation Forest, e x3= SVM One-Class.

- $K = \{k1, k2, k3, k4\}$, onde k representa o tipo de erro injetado na sequência (k1 = Bias, k2 = Freezing, k3 = Loss of Accuracy, e k4 = Noise).

Figura 8 – Erro de Freezing.



Fonte: Produzida pela autora.

- ❏ *N = {n1, n2}*, onde n1 e n2 representam 10 e 20 por cento (%) de dados incorretos injetados na sequência de dados do sensor analisado.

Assim, para um sensor Y contabilizam-se 24 execuções. Então, para y = 30, obtiveram-se 720 execuções.

5.2.12 Etapa 4: Processar Algoritmos e Calcular Métricas de Avaliação

Nesta etapa foi realizado o processamento dos três algoritmos e o cálculo de métricas de avaliação para cada grupo experimental. O algoritmo Isolation Forest e algoritmo SVM One-Class foram aplicados para detectar anomalias nos dois lotes de dados com erros injetados. Em particular, a técnica de Diversidade foi aplicada também para detectar anomalias nos dois lotes de dados. Contudo, devido a característica da técnica de Diversidade, os lotes de dados iniciais, ainda sem erros injetados, também foram utilizados na análise. Isso foi realizado, para que a adição de erros na sequência não influenciasse no desempenho de detecção pelo algoritmo.

5.2.13 Métricas de Avaliação do Experimento

Um sistema de detecção pode ser medido de acordo com a medida F1-Measure (CHRISTEN; HAND; KIRIELLE, 2023) para verificar seu desempenho na classificação ou detecção. Após a etapa de processamento dos algoritmos, os resultados foram avaliados utilizando-se as medidas de Precisão (Precision) e Revocação (Recall), as quais são posteriormente utilizadas no cálculo de F1-Measure. Quanto mais próximas ambas as métricas

estiverem do valor 1, melhor será a eficácia do sistema de detecção. Tais métricas são definidas conforme as Equações 8 e 9.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{8}$$

$$\text{Revocação} = \frac{VP}{VP + FN} \tag{9}$$

Nessas equações, VP (verdadeiro positivo) é o número de erros corretamente detectados pela técnica; FP (falso positivos) é o número de erros detectados pela técnica que não correspondem a erros reais (um dado normal foi classificado erroneamente como anômalo), e FN (falso negativos) é número de erros que deixaram de ser detectados pela técnica. A medida F1-Measure, apresentada na equação 10, corresponde à média harmônica entre a Precisão e a Revocação, e é definida de acordo com a Equação 10.

$$F_1 = 2 \times \frac{Precisão \times Revocação}{Precisão + Revocação} \tag{10}$$

Para o cálculo dessas métricas, os números de VPs, FPs e FNs foram determinados utilizando-se os dados dos sensores marcados automaticamente, ou seja, aqueles dados cujos erros foram injetados e foram automaticamente rotulados. Para um determinado grupo de execução G, calculou-se uma média F1-Measure para os 30 sensores com base na detecção de erros nas 8.640 leituras correspondentes a um dia completo de leituras. Em seguida, foi calculada a média de F1-Measure utilizando-se as medidas F1-Measure de cada sensor para cada grupo de execução.

Além desta métrica, calculou-se uma medida de tempo de processamento em segundos, ou seja, o tempo utilizado no processamento do grupo de execução para cada sensor. Mais uma vez, calculou-se a média do tempo de processamento com base nos 30 sensores. Ao final, foram realizados testes estatísticos adequados, sendo eles: teste de Shapiro-Wilk (SHAPIRO; WILK, 1965) para verificar se as distribuições de dados são (ou não) normais, e os testes T-Student e Mann-Whitney para verificar se há diferenças estatisticamente relevantes entre os resultados obtidos com as diferentes técnicas de detecção de erros em dados de sensores.

5.2.14 Bibliotecas, Hardware e Parâmetros

Os experimentos foram realizados com as sequintes configurações: processador 12th Gen Intel(R) Core(TM) i7-12700T com velocidade de 4.7 GHz, memória volátil igual a 15 GB e sistema operacional equivalente a Ubuntu 20.04.6 LTS. Utilizou-se a IDE PyCharm Community Edition na versão 2024.2.4 e a linguagem de programação Python 3.8.10. Os dados foram salvos em arquivos em formato CSV disponíveis online [1]. Para

[1] https://github.com/Kathiani/Data-Validation-MidVaL-PythonV - acessado em Dezembro de 2024

a implementação dos algoritmos de detecção de anomalias, foram utilizadas as seguintes bibliotecas:

- Algoritmo Isolation Forest: utilizou-se a biblioteca scikit-learn, com o módulo sklearn.ensemble e função IsolationForest.
- Algoritmo SVM One-Class: utilizou-se a biblioteca scikit-learn, com o módulo sklearn.ensemble e função OneClassSVM.

Os parâmetros do algoritmo Isolation Forest foram ajustados para os seguintes valores:

- Número de estimadores (*n-estimators*): o número de árvores afeta a robustez e a precisão do modelo. Mais árvores podem aumentar a estabilidade do modelo, mas geram maior custo computacional. Neste trabalho, o número de estimadores foi fixado em 100.
- Número de amostras (*max-samples*): determina a quantidade de dados usada para construir cada árvore, impactando o equilíbrio entre generalização e especialização. Neste trabalho, o número de amostras foi padrão (isto, é *auto*).
- Contaminação (*contamination*): o limiar usado para classificar um ponto como anômalo. Determina-se com base em estimativas do domínio do problema (ex.: 0.1 para 10% dos dados anômalos).
- Número de características (*max-features*): geralmente é mantido constante, pois costuma ser uma proporção padrão do total de características. Neste trabalho o número de características foi fixado em 1.
- Aleatoriedade (*random-state*): não influencia diretamente o desempenho do modelo; apenas controla a forma como as amostras e divisões são realizadas. Neste trabalho a aleatoriedade foi fixada em 42.

Os parâmetros do algoritmo SVM One-Class foram ajustados da seguinte maneira:

- ν (*nu*): a fração máxima de outliers que o modelo pode identificar. Neste trabalho, foi fixado o valor de 0.1, o qual considera o máximo 10% dos dados como anomalias.
- γ (*gamma*): Controla o alcance da função kernel. Neste trabalho, foi fixado o valor 0.1, permitindo maior generalização do modelo.

Tanto para o Isolation Forest quanto para o SVM One-Class, diferentes valores foram testados empiricamente e os resultados avaliados visualmente para um pequeno conjunto de dados.

5.3 Resultados dos Experimentos

5.3.1 F1-Measure e Tempo de Processamento

Nesta seção apresentam-se os resultados das médias de F1-Measure e médias de tempo de processamento dos 30 sensores para para cada grupo de execução para as sequências não-sazonais. Tais resultados são apresentados nas Tabelas 21 e 22, respectivamente. Os resultados são apresentados para todas as técnicas, lotes e tipos de erro avaliados.

Em geral, observa-se que os algoritmos sofrem degradação conforme a quantidade de erros aumenta, ou seja, há diminuição no valor do F1-Measure quando se contrastam os lotes de dados 1 (L1) e os lotes de dados 2 (L2). Na Tabela 21 observa-se que os melhores resultados do F1-Measure, **para cada técnica**, foram: F1-Measure-diversidade-Bias-L1 = 0.9970969683766828, F1-Measure-isolation-Noise-L1 = 0.9318596020944543 e F1-Measure-svm-LossAccuracy-L1 = 0.5361349601791136 (baixa relevância estatística).

Em relação ao tempo de execução, conforme apresentado na Tabela 22, os melhores tempos, **para cada técnica**, foram: F1-Measure-diversidade-Bias-L1 = 0.021610204378763802 segundos, F1-Measure-isolation-LossAccuracy-L1 = 0.0666256745656331 segundos e F1-Measure-svm-LossAccuracy-L1 = 0.29682710965474446 segundos.

Tabela 21 – Média da medida F1-Measure para os 30 sensores.

Id-Grupo-Experimental	Técnica	Lote	Tipo de Erro	Média F1-Measure
F1-Measure-diversidade-Bias-L1	Diversidade	L1	Bias	**0.9970969683766828**
F1-Measure-diversidade-Bias-L2	Diversidade	L2	Bias	0.9982193641996717
F1-Measure-diversidade-Freezing-L1	Diversidade	L1	Freezing	0.0008781854593814201
F1-Measure-diversidade-Freezing-L2	Diversidade	L2	Freezing	0.0012928764938307002
F1-Measure-diversidade-LossAccuracy-L1	Diversidade	L1	LossAccuracy	0.0315503026099831
F1-Measure-diversidade-LossAccuracy-L2	Diversidade	L2	LossAccuracy	0.03124419853817639
F1-Measure-diversidade-Noise-L1	Diversidade	L1	Noise	0.8899305234498815
F1-Measure-diversidade-Noise-L2	Diversidade	L2	Noise	0.8895047285213706
F1-Measure-isolation-Bias-L1	Isolation Forest	L1	Bias	0.4743591036064779
F1-Measure-isolation-Bias-L2	Isolation Forest	L2	Bias	0.4763788808621227
F1-Measure-isolation-Freezing-L1	Isolation Forest	L1	Freezing	0.0
F1-Measure-isolation-Freezing-L2	Isolation Forest	L2	Freezing	0.0
F1-Measure-isolation-LossAccuracy-L1	Isolation Forest	L1	LossAccuracy	0.2066673208525641
F1-Measure-isolation-LossAccuracy-L2	Isolation Forest	L2	LossAccuracy	0.18120041009575075
F1-Measure-isolation-Noise-L1	Isolation Forest	L1	Noise	**0.9318596020944543**
F1-Measure-isolation-Noise-L2	Isolation Forest	L2	Noise	0.6522062204610456
F1-Measure-svm-Bias-L1	SVM	L1	Bias	0.21669099737667893
F1-Measure-svm-Bias-L2	SVM	L2	Bias	0.2598106122913675
F1-Measure-svm-Freezing-L1	SVM	L1	Freezing	0.0
F1-Measure-svm-Freezing-L2	SVM	L2	Freezing	0.0
F1-Measure-svm-LossAccuracy-L1	SVM	L1	LossAccuracy	0.16490386106207167
F1-Measure-svm-LossAccuracy-L2	SVM	L2	LossAccuracy	0.10677017951077107
F1-Measure-svm-Noise-L1	SVM	L1	Noise	**0.5361349601791136**
F1-Measure-svm-Noise-L2	SVM	L2	Noise	0.4003674616087914

5.4 Análise Comparativa dos Resultados

Antes de realizar os testes estatísticos, aplicou-se o teste de Shapiro-Wilk a fim de verificar a normalidade dos dados e selecionar o teste estatístico adequado (isto é, Teste T-Student, ou Teste Mann-Whitney) a ser utilizado entre os grupos de execução comparados.

Tabela 22 – Média de tempo de processamento em segundos para o 30 sensores.

Id-Grupo-Experimental	Técnica	Lote	Tipo de Erro	Tempo Médio
F1-Measure-diversidade-Bias-L1	Diversidade	L1	Bias	**0.021610204378763802**
F1-Measure-diversidade-Bias-L2	Diversidade	L2	Bias	0.022868982950846307
F1-Measure-diversidade-Freezing-L1	Diversidade	L1	Freezing	0.0227034489313761
F1-Measure-diversidade-Freezing-L2	Diversidade	L2	Freezing	0.022207887967427512
F1-Measure-diversidade-LossAccuracy-L1	Diversidade	L1	LossAccuracy	0.022726639111836703
F1-Measure-diversidade-LossAccuracy-L2	Diversidade	L2	LossAccuracy	0.02327689329783117
F1-Measure-diversidade-Noise-L1	Diversidade	L1	Noise	0.02231901486714676
F1-Measure-diversidade-Noise-L2	Diversidade	L2	Noise	0.024946610132853143
F1-Measure-isolation-Bias-L1	Isolation Forest	L1	Bias	0.0797610521316528
F1-Measure-isolation-Bias-L2	Isolation Forest	L2	Bias	0.0750031232833862
F1-Measure-isolation-Freezing-L1	Isolation Forest	L1	Freezing	0.07527335484822588
F1-Measure-isolation-Freezing-L2	Isolation Forest	L2	Freezing	0.06789490381876624
F1-Measure-isolation-LossAccuracy-L1	Isolation Forest	L1	LossAccuracy	**0.0666256745656331**
F1-Measure-isolation-LossAccuracy-L2	Isolation Forest	L2	LossAccuracy	0.0760509093602498
F1-Measure-isolation-Noise-L1	Isolation Forest	L1	Noise	0.0778218269348144
F1-Measure-isolation-Noise-L2	Isolation Forest	L2	Noise	0.0807649453481038
F1-Measure-svm-Bias-L1	SVM	L1	Bias	0.30560033321380614
F1-Measure-svm-Bias-L2	SVM	L2	Bias	0.3512544552485148
F1-Measure-svm-Freezing-L1	SVM	L1	Freezing	0.29520126183827716
F1-Measure-svm-Freezing-L2	SVM	L2	Freezing	0.32317932446797687
F1-Measure-svm-LossAccuracy-L1	SVM	L1	LossAccuracy	**0.29682710965474446**
F1-Measure-svm-LossAccuracy-L2	SVM	L2	LossAccuracy	0.3039791186650594
F1-Measure-svm-Noise-L1	SVM	L1	Noise	0.30586853822072346
F1-Measure-svm-Noise-L2	SVM	L2	Noise	0.495893661181132

A seguir são apresentados os testes estatísticos para as hipóteses para taxa de detecção de erros e tempo de processamento.

5.4.1 Testes Estatísticos sobre Hipóteses para Taxa de Detecção

Nesta seção, são apresentados os resultados dos testes estatísticos para as médias F1-Measure. Os resultados para as hipóteses H1 e H2 para o lote L1 são apresentados nas tabelas 23 e 24, respectivamente. Analogamente, os resultados para as hipóteses H1 e H2 para o lote L1 são apresentados nas tabelas 25 e 26, respectivamente. Em todas essas tabelas, as siglas M e TS representam os testes estatísticos realizados, Mann-Whitney e Teste T-Student, respectivamente. As siglas Av., Dif. e Ig. significam Avaliação do Teste, Diferentes Distribuições e Distribuições Iguais, respectivamente. Nas tabelas os melhores resultados são apresentados em negrito.

5.4.2 Testes Estatísticos sobre Hipóteses para Tempo de Processamento

Nesta seção são apresentados os resultados dos testes estatísticos para as médias de tempo de processamento. Os resultados para as hipóteses H3 e H4 para o lote L1 são apresentados nas tabelas 27 e 29, respectivamente. Analogamente, os resultados para as hipóteses H3 e H4 para o lote L2 são apresentados nas tabelas 28 e 30, respectivamente. Os melhores resultados aparecem destacados em negrito nas tabelas.

Tabela 23 – Testes para a H1 para o lote L1.

Id do Grupo Experimental	Técnica	Erro	F1-Measure	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L1	Isolation Forest	Bias	0.4743591036064779	M		
F1-Measure-svm-Bias-L1	SVM		0.21669099737667893		1.7807929095681583e-24	Ig.
F1-Measure-isolation-Freezing-L1	Isolation Forest	Freezing	0.0	M		
F1-Measure-svm-Freezing-L1	SVM		0.0		NaN	Ig.
F1-Measure-isolation-LossAccuracy-L1	Isolation Forest	LossAccuracy	0.2066673208525641	M		
F1-Measure-svm-LossAccuracy-L1	SVM		0.16490386106207167		0.004320638881059	Dif.
F1-Measure-isolation-Noise-L1	Isolation	Noise	0.9318596020944543	M		
F1-Measure-svm-Noise-L1	SVM		0.5361349601791136		0.000000055094901	Dif.

Tabela 24 – Testes para a H2 para o lote L1.

Id do Grupo Experimental	Técnica	Erro	F1-Measure	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L1	Isolation Forest	Bias	0.4743591036064779	TS		
F1-Measure-diversidade-Bias-L1	Diversidade		0.9970969683766828		5.530643719901697e-78	Dif
F1-Measure-svm-Freezing-L1	Isolation Forest	Freezing	0.0	M		
F1-Measure-diversidade-Freezing-L1	Diversidade		0.0008781854593814201		0.000065918180122	Dif
F1-Measure-Isolation-LossAccuracy-L1	Isolation Forest	Loss of Accuracy	0.2066673208525641	M		
F1-Measure-diversidade-LossAccuracy-L1	Diversidade		0.0315503026099831		0.000001434205044	Dif
F1-Measure-isolation-Noise-L1	Isolation Forest	Noise	0.9318596020944543	TS		
F1-Measure-diversidade-Noise-L1	Diversidade		0.8899305234498815		3.5401059418831l5e-30	Dif

Tabela 25 – Testes para a H1 para o lote L2.

Id do Grupo Experimental	Técnica	Erro	F1-Measure	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L2	Isolation Forest	Bias	0.4763788808621227	M		
F1-Measure-svm-Bias-L2	SVM		0.2598106122913675		0.000000010982304	Dif.
F1-Measure-isolation-Freezing-L2	Isolation Forest	Freezing	0.0	TS		
F1-Measure-svm-Freezing-L2	SVM		0.0		NaN	Ig.
F1-Measure-isolation-LossAccuracy-L2	Isolation Forest	Loss of Accuracy	0.2066673208525641	M		
F1-Measure-svm-LossAccuracy-L2	SVM		0.10677017951077107		0.000003834942359	Dif.
F1-Measure-isolation-Noise-L2	Isolation Forest	Noise	0.6522062204610456	M		
F1-Measure-svm-Noise-L2	SVM	Noise	0.4003674616087914		0.000178133562050	Dif.

Tabela 26 – Testes para a H2 para o lote L2.

Id do Grupo Experimental	Técnica	Erro	F1-Measure	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L2	Isolation Forest	Bias	0.4763788808621227	TS		
F1-Measure-diversidade-Bias-L2	Diversidade		0.9982193641996717		3.9304615964111444e-82	Dif.
F1-Measure-Isolation-Freezing-L2	Isolation Forest	Freezing	0.0	TS		
F1-Measure-Diversidade-Freezing-L2	Diversidade		0.0012928764938307002		0.000029179689431	Dif.
F1-Measure-Isolation-LossAccuracy-L2	Isolation Forest	Loss of Accuracy	0.18120041009575075	M		
F1-Measure-diversidade-LossAccuracy-L2	Diversidade		0.03124419853817639		0.000008877782891	Dif.
F1-Measure-isolation-Noise-L2	Isolation Forest	Noise	0.6522062204610456	TS		
F1-Measure-diversidade-Noise-L2	Diversidade		0.8895047285213706		1.001942280168709e-73	Dif.

Tabela 27 – Testes para a H3 para o lote L1.

Id do Grupo Experimental	Técnica	Erro	Tempo Médio	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L1	Isolation Forest	Bias	0.0797610521316528	M		
F1-Measure-svm-Bias-L1	SVM		0.30560033321380614		0.000000000030199	Dif.
F1-Measure-isolation-Freezing-L1	Isolation Forest	Freezing	0.07527335484822588	M		
F1-Measure-svm-Freezing-L1	SVM		0.29520126183827716		0.000000000030199	Dif.
F1-Measure-isolation-LossAccuracy-L1	Isolation Forest	Loss of Accuracy	0.0666256745656331	M		
F1-Measure-svm-LossAccuracy-L1	SVM		0.29682710965474446		0.000000000030199	Dif.
F1-Measure-isolation-Noise-L1	Isolation Forest	Noise	0.0778218269348144	M		
F1-Measure-svm-Noise-L1	SVM		0.30586853822072346		0.000000000030199	Dif.

5.5 Respostas às Hipóteses e Conclusões do Experimento

As respostas às hipóteses definidas para o experimento relatado neste capítulo, baseando-se nos resultados para as sequências não-sazonais, são exibidas nas Tabelas 31, 32, 33, e 34. Nas tabelas, a sigla L1 significa lote de dados 1, enquanto L2 significa lote de dados 2. Em adição, as técnicas que aparecem destacadas em vermelho indicam o ganho com baixa relevância estatística.

Observando-se a Tabela de análise 31, percebe-se que para os erros de Bias e Noise, o algoritmo Isolation Forest apresentou maior taxa de detecção para todos os erros em

Tabela 28 – Testes para a H3 para o lote L2.

Id do Grupo Experimental	Técnica	Erro	Tempo Médio	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L2	Isolation Forest	Bias	0.0750031232833862	M		
F1-Measure-svm-Bias-L2	SVM		0.3512544552485148		0.000000000030199	Dif.
F1-Measure-isolation-Freezing-L2	Isolation Forest	Freezing	0.06789490381876624	M		
F1-Measure-svm-Freezing-L2	SVM		0.32317932446797687		0.000000000030199	Dif.
F1-Measure-isolation-LossAccuracy-L2	Isolation Forest	Loss of Accuracy	0.0760509093602498	M		
F1-Measure-svm-LossAccuracy-L2	SVM		0.3039791186650594		0.000000000030199	Dif.
F1-Measure-isolation-Noise-L2	Isolation Forest	Noise	0.0807649453481038	M		
F1-Measure-svm-Noise-L2	SVM		0.495893661181132		0.000000000030199	Dif.

Tabela 29 – Testes para a H4 para o lote L1.

Id do Grupo Experimental	Técnica	Erro	Tempo Médio	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L1	Isolation Forest	Bias	0.0797610521316528	M		
F1-Measure-diversidade-Bias-L1	Diversidade		0.021610204378763802		0.000000000030199	Dif.
F1-Measure-isolation-Freezing-L1	Isolation Forest	Freezing	0.07527335484822588	M		
F1-Measure-diversidade-Freezing-L1	Diversidade		0.0227034489313761		0.000000000030199	Dif.
F1-Measure-isolation-LossAccuracy-L1	Isolation Forest	Loss of Accuracy	0.0666256745656331	M		
F1-Measure-diversidade-LossAccuracy-L1	Diversidade		0.022726639111836703		0.000000000030199	Dif.
F1-Measure-isolation-Noise-L1	Isolation Forest	Noise	0.0778218269348144	M		
F1-Measure-diversidade-Noise-L1	Diversidade		0.02231901486714676		0.000000000030199	Dif.

todos os lotes. Observando-se a Tabela de análise 32, ao comparar os melhores algoritmos de detecção de anomalias com o algoritmo Diversidade por correlação, nota-se que Diversidade apresenta os melhores resultados para os erros de Bias e Freezing tanto para L1 quanto para L2. O algoritmo SVM One-Class apresentou os piores resultados para todos os lotes para o caso das métricas de F1-measure.

Ao observar-se as médias da métrica F1-Measure, constata-se que ao aumentar a porcentagem de erros, os algoritmos de detecção de anomalias diminuem sua capacidade de detecção. Com base nessa observação, também é possível notar que os algoritmos de detecção de erros perdem espaço para Diversidade. Com relação às hipóteses para tempo de processamento (H3 e H4 - vide Tabelas 33 e 34), o Isolation Forest exibiu menor tempo de processamento quando comparado ao SVM One-Class. Em seguida ao

Tabela 30 – Testes para a H4 para o lote L2.

Id do Grupo Experimental	Técnica	Erro	Tempo Médio	Teste	P-Value	Av.
F1-Measure-isolation-Bias-L2	Isolation Forest	Bias	0.0750031232833862	M		
F1-Measure-diversidade-Bias-L2	Diversidade		0.022868982950846307		0.000000000030180	Dif.
F1-Measure-Isolation-Freezing-L2	Isolation Forest	Freezing	0.06789490381876624	M		
F1-Measure-diversidade-Freezing-L2	Diversidade		0.022207887967427512		0.000000000030199	Dif.
F1-Measure-Isolation-LossAccuracy-L2	Isolation Forest	Loss of Accuracy	0.0760509093602498	M		
F1-Measure-diversidade-LossAccuracy-L2	Diversidade		0.02327689329783117		0.000000000030199	Dif.
F1-Measure-isolation-Noise-L2	Isolation Forest	Noise	0.0807649453481038	M		
F1-Measure-diversidade-Noise-L2	Diversidade		0.024946610132853143		0.000000000030199	Dif.

comparar Isolation com Diversidade, esta última apresenta o menor tempo. Diversidade destacou-se para ambas as medidas (tempo e detecção), enquanto bons resultados foram apresentados para a técnica Isolation Forest principalmente para o erro de Noise.

Tabela 31 – Respostas para a hipótese H1 (fixando conjunto de técnicas). Em vermelho é apresentada a melhor métrica F1-Measure.

Tipo de Erro	**Hipótese H1**	L1	L2
Bias	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
Freezing	Sim, há diferença, hipótese nula rejeitada	**Nenhum**	**Nenhum**
LossAccuracy	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
Noise	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**

Tabela 32 – Respostas para a hipótese H2 (fixando conjunto de técnicas). Em vermelho é apresentada a melhor métrica F1-Measure.

Tipo de Erro	**Hipótese H2**	L1	L2
Bias	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**
Freezing	Sim, há diferença, hipótese nula rejeitada	**Nenhum**	**Nenhum**
LossAccuracy	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
Noise	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**

Tabela 33 – Respostas para a hipótese H3 (fixando conjunto de técnicas). Em vermelho é apresentada a melhor (menor) métrica de tempo.

Tipo de Erro	**Hipótese H3**	L1	L2
Bias	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
Freezing	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
LossAccuracy	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**
Noise	Sim, há diferença, hipótese nula rejeitada	**Isolation**	**Isolation**

Tabela 34 – Respostas para a hipótese H4 (fixando conjunto de técnicas). Em vermelho é apresentada a melhor (menor) métrica de tempo.

Tipo de Erro	**Hipótese H4**	L1	L2
Bias	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**
Freezing	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**
LossAccuracy	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**
Noise	Sim, há diferença, hipótese nula rejeitada	**Diversidade**	**Diversidade**

5.6 Ameaças à Validade

As ameaças à validade do experimento são:

- Validade interna: está relacionada ao quanto os resultados observados refletem a realidade dos experimentos. Neste trabalho, o ajuste de parâmetros dos algoritmos foi específico para um cenário. Desta forma, os resultados podem não ser generalizáveis para outros contextos de Cidades Inteligentes.

- Validade Externa: está relacionada ao fato da amostra não ser representativa da população. Neste experimento foi realizada a simulação de dados de Aplicações para Cidades Inteligentes para dados de sensores de temperatura. No entanto, outros cenários devem ser testados para avaliar o desempenho na detecção de erros pelos algoritmos em diferentes cenários. Outros exemplos que poderiam servir na avaliação são aplicações que contam a quantidade de carros nas ruas e atualizam semáforos inteligentes e aplicações que medem o nível de água de rios que entrecortam cidades em situações de enchentes.

5.7 Considerações Finais

Este capítulo apresentou o estudo experimental conduzido a fim de avaliar algoritmos de detecção de anomalias no contexto de dados de Aplicações para Cidades Inteligentes. Ao final de tal avaliação, obtiveram-se as conclusões para a escolha do melhor algoritmo de detecção em termos de taxa de detecção de erros e tempo de processamento para diferentes tipos de erro. Dadas as análises, Diversidade exibiu melhor taxa de detecção e desempenho para o pior cenário de erros. Com tais conclusões, torna-se possível a viabilidade de utilização dos algoritmos testados para compor um serviço de validação de sensores de Aplicações para Cidades Inteligentes, conforme abordagem apresentada no Capítulo 4.

Para o experimento, foram criados diferentes cenários de erros e também um teste para um sequência sazonal. Foram realizadas análises estatísticas e teste de hipóteses a fim de verificar a relevância estatística das diferenças entre os algoritmos. Os resultados de detecção são promissores para a abordagem de validação de dados de Aplicações para Cidades Inteligentes apresentada nesta tese. Para os casos de detecção em sensores isolados, ou seja, onde não há uma quantidade grande dos mesmos sensores disponíveis, os

algoritmos de detecção de anomalias podem ser utilizados, com destaque para Isolation Forest, que apresenta taxas de detecção promissoras e com menor tempo de processamento em comparação com o algoritmo SVM One-Class.

No próximo capítulo conclui-se esta tese, retomando-se o contexto e resumindo-se as contribuições obtidas, juntamente com as limitações do trabalho, possibilidades de trabalhos futuros, e publicações derivadas deste trabalho.

Capítulo 6

Conclusão

Nesta tese, primeiramente realizou-se uma revisão sistemática da literatura sobre tolerância a defeitos em sistemas adaptativos e sensíveis ao contexto, a fim de identificar as técnicas investigadas e/ou aplicadas para esse tipo de sistemas, e quais os defeitos, erros e falhas que estiveram em foco pela comunidade de pesquisa. Em seguida, com as técnicas de tolerância a defeitos elencadas e uma base de conhecimento levantada, foi realizada uma revisão sistemática da literatura sobre tolerância a defeitos em Aplicações para Cidades Inteligentes que possuem a propriedade de ciência de contexto em seu comportamento. Com base nos resultados coletados da revisão, concebeu-se uma abordagem para validação e recuperação de erros em dados de sensores de Aplicações para Cidades Inteligentes. Em seguida, foi identificado um conjunto de algoritmos a serem utilizados na etapa de validação de dados, no escopo da abordagem definida. Por fim, uma abordagem experimental foi criada para avaliar os algoritmos investigados.

As contribuições obtidas com o trabalho realizado, e as respectivas limitações e possibilidade de trabalhos futuros são apresentados na sequência deste capítulo, seguidas pela lista de publicações finalizadas e futuras.

6.1 Contribuições da Tese

As principais contribuições desta tese são:

- A definição de uma abordagem para validação e recuperação de erros em dados de sensores de Aplicações para Cidades Inteligentes.

- Uma abordagem experimental para avaliar algoritmos para validação de dados de

sensores. Tal abordagem permite realizar novos estudos, considerando novos algoritmos e novos cenários.

- Duas revisões sistemáticas da literatura contendo direções de pesquisa sobre técnicas de tolerância a defeitos.

6.2 Resultados Obtidos

Os principais resultados alcançados foram:

- A concepção de uma abordagem conceitual para validação e recuperação de erros em dados de sensores para Aplicações para Cidades Inteligentes, como uma implementação preliminar de um middleware que incorpora a abordagem definida.
- A experimentação e avaliação de algoritmos de detecção de anomalias em cenários de Cidades Inteligentes, permitindo sua replicação em outros cenários de Aplicações para Cidades Inteligentes.
- A identificação de um conjunto de técnicas de tolerância a defeitos, e de tipos de defeitos, erros e falhas para Aplicações para Cidades Inteligentes, obtidos por meio da revisão sistemática.
- A identificação de um conjunto de técnicas de tolerância a defeitos e tipos de defeitos para sistemas adaptativos e sistemas sensíveis ao contexto, obtidos por meio da revisão sistemática.

6.3 Limitações do Trabalho

O trabalho proposto nesta tese apresenta as seguintes limitações:

- A definição de parâmetros dos algoritmos foi realizada manualmente. No entanto, existem técnicas que permitem um melhor ajuste de parâmetros por meio de buscas exaustivas ou seleção aleatória de combinações de parâmetros. A utilização dessas técnicas poderia ter maximizado o desempenho de detecção dos algoritmos.
- Para a técnica de diversidade, foi fixado um valor de limiar de comparação igual a 3. Variações para o limiar desta técnica não foram testadas nos experimentos, o que pode alterar o desempenho do algoritmo em cenários de aplicações nos quais o intervalo de valores de um sensor para o outro pode ser maior.
- Técnicas baseadas em regras, assim como as apresentadas pelo estado da arte, não foram comparadas com os algoritmos de detecção de erros propostos. Deste modo, há uma lacuna com relação à comparação de técnicas utilizadas no estado da arte e

as proposta nesta tese. Contudo, realizar uma análise automática apresenta a vantagem de dispensar a análise de cada cenário pelo especialista para definir descrições para os sensores.

- O middleware proposto não teve sua implementação finalizada com os algoritmos de validação e teste em cenários distintos. Deste modo, a avaliação da implementação contendo a abordagem de validação em um cenário real se tornou restrita.

- Os dados utilizados no experimento foram obtidos por meio de uma simulação com o objetivo de obter mais controle sobre o tipo de dado manipulado, a frequência de obtenção e a quantidade de dados simulados, e, além disso, obter dados para experimentação mais rapidamente. No entanto, utilizar dados reais pode garantir que os dados sejam um reflexo de condições reais de operação do sensor, incluindo variações sazonais, interferências do ambiente nos dados e erros reais presentes em dados de sensores. Com dados reais, a avaliação dos algoritmos de detecção pode se tornar mais confiável. Simulações com bases de dados reais serão realizadas em trabalhos futuros.

- Este trabalho teve como alvo inicial da abordagem a plataforma InterSCity. No entanto, os dados utilizados no experimento foram simulados, ou seja, não foram obtidos de nenhuma base específica. Deste modo, deve-se considerar a adequação da abordagem de validação para ser utilizada em dados da InterSCity e também em outras plataformas de Aplicações para Cidades Inteligentes.

6.4 Trabalhos Futuros

Como recomendações de trabalhos futuros, propõe-se:

- Avaliar os algoritmos de detecção em outros cenários de Cidades Inteligentes, de modo a aumentar a garantia de capacidade de detecção em múltiplos cenários.

- Comparar os algoritmos de detecção de erros com as abordagens baseadas em regras e descrições dos sensores mencionadas no estado da arte a fim de verificar ganhos e perdas.

- Realizar o estudo de estratégias adequadas para a recuperação de erros, com a proposta de recuperar dados de sensores considerados incorretos, ou seja, inválidos.

- Construir uma abordagem experimental para avaliar estratégias de recuperação de erros em cenários de Aplicações para Cidades Inteligentes.

- Evoluir o middleware (MIDVAL) que foi preliminarmente implementado durante este trabalho. Nota-se que há um grande potencial para expandir e melhorar os

resultados à medida que a implementação avançar. Sendo assim, um dos trabalhos futuros é continuar o desenvolvimento e adequar a implementação do middleware proposto, que está em uma versão prelimiar. Conforme dito em capítulos anteriores desta tese, uma implementação inicial foi realizada[1]. A implementação foi realizada na linguagem de programação Java, mas deve ser ajustada para aderir aos algoritmos de validação experimentados que utilizam bibliotecas da linguagem de programação Python. Além desta adequação, a implementação inicial precisa de diversos ajustes para validar diferentes dados na InterSCity. Atualmente a validação é feita para determinada capacidade (do inglês, *capability*, conforme conceitos definidos e implementados na Plataforma InterSCity) de um sensor específico previamente cadastrada. A versão inicial está implementada em Java 17 e com o framework Spring Boot 3. Esta implementação inicial obtém dados previamente cadastrados da plataforma InterSCity, e os valida por meio de análise de atualidade do dado, ou seja, verifica se o dado foi salvo na base em um determinado tempo limite. Nesta implementação também aplica-se uma nova tentativa de obtenção do dado (correspondente as etapas 2 e 3 de validação mostradas na Figura 3) e recupera-se o dado com base na média histórica.

6.5 Lista de Publicações

Os trabalhos em colaboração publicados, em primeira autoria, estão descritos abaixo:

- K. E. De Souza and F. C. Ferrari: "A Systematic Review of Fault Tolerance Techniques for Adaptive and Context-Aware Systems", IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), CA, USA, 2022, pp. 21-30, doi: <https://doi.org/10.1109/ACSOS55765.2022.00020>

- K. E. de Souza, F. C. Ferrari, V. V. de Camargo, M. Ribeiro, J. Offutt: "A systematic review of fault tolerance techniques for smart city applications", Journal of Systems and Software, Volume 219, N. 112249, 2025, doi: <https://doi.org/10.1016/j.jss.2024.112249>

Os trabalhos em colaboração publicados estão descritos abaixo:

- B. R. Siqueira, F. C. Ferrari, K. E. Souza, D. S. M. Santibáñez and V. V. Camargo,"Faults Types of Adaptive and Context-Aware Systems and Their Relationship with Fault-based Testing Approaches," The 15th International Workshop on Mutation Analysis (Mutation), 2020, pp. 284-293, doi: <https://doi.org/10.1109/ICSTW50294.2020.00054>

[1] <https://github.com/Kathiani/MidVAL-SensorValidationAndRecovery> e está disponível online

- ❑ B. R. Siqueira, F. C. Ferrari, K. E. Souza, V. V. Camargo, R. de Lemos: Testing of Adaptive and Context-aware Systems: Approaches and Challenges. Software Testing, Verification and Reliability, Volume 31 (7), e1772, 2021. doi: <https://doi.org/10.1002/stvr.1772>

O trabalho futuro a ser publicado, derivado desta tese, refere-se ao design experimental e resultados obtidos na detecção de dados incorretos.

Referências

ABREU, D. P. et al. A resilient internet of things architecture for smart cities. **Annals of Telecommunications**, Springer-Verlag, v. 72, p. 19–30, 2017.

AHMAD, S. et al. Complex problems solution as a service based on predictive optimization and tasks orchestration in smart cities. **Computers, Materials & Continua**, v. 69, n. 1, p. 1271–1288, 2021.

AHMED, M.; MAHMOOD, A. N.; ISLAM, M. R. A survey of anomaly detection techniques in financial domain. **Future Generation Computer Systems**, v. 55, p. 278–288, 2016.

AHMED, M.; Naser Mahmood, A.; HU, J. A survey of network anomaly detection techniques. **Journal of Network and Computer Applications**, v. 60, p. 19–31, 2016.

ALAG, S.; AGOGINO, A.; MORJARIA, M. A methodology for intelligent sensor measurement, validation, fusion, and fault detection for equipment monitoring and diagnostics. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing**, v. 15, 2001.

ALI, H.; SOE, J. K.; WELLER, S. R. A real-time ambient air quality monitoring wireless sensor network for schools in smart cities. In: **Proceedings of the** 1^{th} **IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2015. p. 1–6.

ALI, J. et al. Mathematical modeling and validation of retransmission-based mutant mqtt for improving quality of service in developing smart cities. **Sensors**, v. 22, n. 24, p. 9751, 2022.

ALI, J.; ZAFAR, M. H. Improved end-to-end service assurance and mathematical modeling of message queuing telemetry transport protocol based massively deployed fully functional devices in smart cities. **Alexandria Engineering Journal**, v. 72, p. 657–672, 2023.

ALJOHANI, S. L.; ALENAZI, M. J. F. Mpresisdn: Multipath resilient routing scheme for SDN-enabled smart cities networks. **Applied Sciences**, v. 11, n. 4, 2021.

ALKADY, G. I. et al. Reliable fpga-based network architecture for smart cities. In: **Proceedings of the** 31^{th} **International Conference on Microelectronics (ICM)**. New York, USA: IEEE, 2019. p. 334–337.

ALZOMAN, R.; ALENAZI, M. J. Exploiting sdn to improve qos of smart city networks against link failures. In: **Proceedings of the 7^{th} International Conference on Software Defined Systems (SDS)**. New York, USA: IEEE, 2020. p. 100–106.

AMMANN, P.; OFFUTT, J. **Introduction to Software Testing**. 2nd. ed. Cambridge, UK: Cambridge University Press, 2017. ISBN 978-1107172012.

ANUPONG, W. et al. Towards a high precision in ami-based smart meters and new technologies in the smart grid. **Sustainable Computing: Informatics and Systems**, v. 35, 2022.

ARASTEH, H. et al. Iot-based smart cities: A survey. In: **2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)**. [S.l.: s.n.], 2016. p. 1–6.

AVIZIENIS, A. Design of fault-tolerant computers. In: **Proceedings of the Fall Joint Computer Conference**. New York, USA: Association for Computing Machinery, 1967. p. 733–743.

_____. The n-version approach to fault-tolerant software. **IEEE Transactions on Software Engineering**, SE-11, n. 12, p. 1491–1501, 1985.

AVIZIENIS, A. et al. Basic concepts and taxonomy of dependable and secure computing. **IEEE Transactions on Dependable and Secure Computing**, IEEE Computer Society Press, v. 1, n. 1, p. 11–33, 2004.

BANDYOPADHYAY, S. et al. Role of middleware for internet of things: A study. **International Journal of Computer Science & Engineering Survey**, v. 2, 2011.

BAROFFIO, L. et al. A visual sensor network for parking lot occupancy detection in smart cities. In: **2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)**. [S.l.: s.n.], 2015. p. 745–750.

BASILI, V. R. **Software modeling and measurement: the Goal/Question/Metric paradigm**. USA, 1992.

BAWANY, N.; SHAMSI, J. Seal: Sdn based secure and agile framework for protecting smart city applications from ddos attacks. **Journal of Network and Computer Applications**, Elsevier, Netherlands, v. 145, p. 102381, 2019.

Beder, D. M.; Araujo, R. B. Towards the definition of a context-aware exception handling mechanism. In: **Proceedings of 5^{th} Latin-American Symposium on Dependable Computing Workshops (LADC)**. New York, USA: IEEE, 2011. p. 25–28.

BIDI, A. H.; MOVAHEDI, Z.; MOVAHEDI, Z. A fog-based fault-tolerant and qoe-aware service composition in smart cities. **Transactions on Emerging Telecommunications Technologies**, v. 32, n. 11, p. e4326, 2021.

BOURQUE, P.; FAIRLEY, R. (Ed.). **Guide to the Software Engineering Body of Knowledge, Version 3.0**. New York, USA: IEEE, 2014.

BRANISAVLJEVIC, N.; KAPELAN, Z.; PRODANOVIC, D. Improved real-time data anomaly detection using context classification. **Journal of Hydroinformatics**, v. 13, n. 3, 2011.

Cai, H.; Peng, C.; Zhang, Y. A novel framework of self-adaptive fault-tolerant for pervasive computing. In: **Proceedings of the 6^{th} International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)**. New York, USA: IEEE, 2012. p. 286–291.

CAPRARESCU, B. A. Robustness and scalability: A dual challenge for autonomic architectures. In: **Proceedings of the 4^{th} European Conference on Software Architecture: Companion Volume (ECSA)**. New York, USA: Association for Computing Machinery, 2010. p. 22–26.

CARAGLIU, A.; BO, C. D.; NIJKAMP, P. Smart cities in europe. **Journal of Urban Technology**, Routledge, v. 18, n. 2, p. 65–82, 2011.

CHELLOUG, S.; EL-ZAWAWY, M. Middleware for internet of things: Survey and challenges. **Intelligent Automation and Soft Computing**, v. 24, p. 1–9, 02 2017.

CHEN, C.; YE, C.; JACOBSEN, H. Hybrid context inconsistency resolution for context-aware services. In: **Proceedings of the 27^{th} International Conference on Pervasive Computing and Communications (PerCom)**. New York, EUA: IEEE, 2011. p. 10–19.

CHENG, B. et al. Building a big data platform for smart cities: Experience and lessons from santander. In: **Proceedings of the 4^{th} International Congress on Big Data**. [S.l.: s.n.], 2015. p. 592–599.

Cho, E.; Helal, S. A situation-based exception detection mechanism for safety in pervasive systems. In: **Proceedings of the International Workshop on Embedded Intelligence and Smart Systems (WEISS)**. New York, USA: IEEE, 2011. p. 196–201.

_____. Toward efficient detection of semantic exceptions in context-aware systems. In: 9^{th} **International Conference on Ubiquitous Intelligence and Computing and 9^{th} International Conference on Autonomic and Trusted Computing**. [S.l.: s.n.], 2012. p. 826–831.

CHRISTEN, P.; HAND, D. J.; KIRIELLE, N. A review of the f-measure: Its history, properties, criticism, and alternatives. Association for Computing Machinery, New York, NY, USA, v. 56, n. 3, out. 2023.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, Sep 1995.

COSTA, F. S.; NASSAR, S. M.; DANTAS, M. A. R. Focuser: A fog online context-aware up-to-date sensor ranking method. **Journal of Sensor and Actuator Networks**, v. 11, n. 2, p. 25, 2022.

CRESTANI, D.; GODARY-DEJEAN, K.; LAPIERRE, L. Enhancing fault tolerance of autonomous mobile robots. **Robotics and Autonomous Systems**, v. 68, p. 140–155, 2015.

CUBO, J.; CANAL, C.; PIMENTEL, E. Model-based dependable composition of self-adaptive systems. **Informatica (Ljubljana)**, v. 35, n. 1, p. 51–62, 2011.

Cubo, J. et al. Reconfiguration of service failures in damasco using dynamic software product lines. In: **Proceedings of the** 11^{th} **International Conference on Services Computing (SCC)**. New York, USA: IEEE, 2015. p. 114–121.

Cui, Y. et al. A new fault tolerance method for field robotics through a self-adaptation architecture. In: **Proceedings of the** 12^{th} **International Symposium on Safety, Security, and Rescue Robotics (SSRR)**. New York, USA: IEEE, 2014. p. 1–6.

DAMASCENO, K. et al. Context-aware exception handling in mobile agent systems: The moca case. In: **Proceedings of the** 5^{th} **International Workshop on Software Engineering for Large-scale Multi-agent Systems (SELMAS)**. New York, USA: Association for Computing Machinery, 2006. p. 37–44.

DAMERI, R. P. Searching for smart city definition: a comprehensive proposal. **INTERNATIONAL JOURNAL OF COMPUTERS AND TECHNOLOGY**, v. 11, n. 5, p. 2544–2551, Oct. 2013.

De Lemos, R. Architectural reconfiguration using coordinated atomic actions. In: **Proceedings of the** 2^{th} **International Workshop on Self-Adaptation and Self-Managing Systems (SEAMS)**. [S.l.: s.n.], 2006. p. 44–50.

de Souza, K. E. et al. A systematic review of fault tolerance techniques for smart city applications. **Journal of Systems and Software**, v. 219, p. 112249, 2025.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. Rio de Janeiro, Brazil: Elsevier, 2007.

den Hamer, P.; Skramstad, T. Autonomic service-oriented architecture for resilient complex systems. In: **Proceedings of the** 30^{th} **Symposium on Reliable Distributed Systems Workshops (SRDS)**. New York, USA: IEEE, 2011. p. 62–66.

DU, R. et al. The sensable city: A survey on the deployment and management for smart city monitoring. **IEEE Communications Surveys & Tutorials**, v. 21, n. 2, p. 1533–1560, 2019.

Ebnenasir, A. Designing run-time fault-tolerance using dynamic updates. In: 2^{th} **International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. New York, USA: IEEE, 2007. p. 15–15.

Eleuterio, J. D. A. S. et al. On the dependability for dynamic software product lines: A comparative systematic mapping study. In: **42nd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)**. [S.l.: s.n.], 2016. p. 323–330.

ESPOSTE, A. M. D. et al. Interscity: A scalable microservice-based open source platform for smart cities. In: **Proceedings of the 6th International Conference on Smart Cities and Green ICT Systems**. [S.l.: s.n.], 2017. p. 35–46.

_____. **InterSCity Platform Architecture**. 2017. <https://gitlab.com/interscity/interscity-platform/docs/-/blob/master/architecture/Architecture.md>. Acessado em: 17/03/2025.

FADHEL, M. A. et al. Comprehensive systematic review of information fusion methods in smart cities and urban environments. **Information Fusion**, v. 107, p. 102317, 2024.

FARIZI, W. S. A.; HIDAYAH, I.; RIZAL, M. N. Isolation forest based anomaly detection: A systematic literature review. In: 8^{th} **International Conference on Information Technology, Computer and Electrical Engineering (ICITACEE)**. [S.l.: s.n.], 2021. p. 118–122.

FERRARI, P. et al. On the use of lorawan and cloud platforms for diversification of mobility-as-a-service infrastructure in smart city scenarios. **IEEE Transactions on Instrumentation and Measurement**, v. 71, p. 1–9, 2022.

FERSI, G. Middleware for internet of things: A study. In: **Proceedings of the** 15^{th} **International Conference on Distributed Computing in Sensor Systems (DCOSS)**. New York, USA: IEEE, 2015. p. 230–235.

Filho, C. Q. et al. Conext-u: A context-aware exception handling mechanism for task-based ubiquitous systems. In: 28^{th} **International Conference on Advanced Information Networking and Applications Workshops (AINA)**. New York, USA: IEEE, 2014. p. 127–132.

GARLAN, D. et al. Rainbow: Architecture-based self-adaptation with reusable infrastructure. **IEEE Computer**, v. 37, n. 10, p. 45–54, 2004.

GARVIN, B. J.; COHEN, M. B.; DWYER, M. B. Failure avoidance in configurable systems through feature locality. In: **Assurances for Self-Adaptive Systems: Principles, Models, and Techniques**. Berlin, Germany: Springer-Verlag, 2013. p. 266–296.

GAY, D.; LEVIS, P.; CULLER, D. Software design patterns for tinyos. **ACM Trans. Embed. Comput. Syst.**, v. 6, n. 4, p. 22–es, 2007.

GEROSTATHOPOULOS, I. et al. Tuning self-adaptation in cyber-physical systems through architectural homeostasis. **Journal of Systems and Software**, v. 148, p. 37–55, 2019.

GHARAIBEH, A. et al. Smart cities: A survey on data management, security, and enabling technologies. **IEEE Communications Surveys & Tutorials**, v. 19, n. 4, p. 2456–2501, 2017.

GIFFINGER, R. et al. **Smart cities - Ranking of European medium-sized cities**. Vienna, Austria: Vienna University of Technology, 2007.

GIRTELSCHMID, S. et al. Big data in large scale intelligent smart city installations. In: **Proceedings of the** 15^{th} **International Conference on Information Integration and Web-Based Applications & Services (IIWAS)**. New York, USA: Association for Computing Machinery, 2013. p. 428–432.

GORLA, A. et al. Achieving cost-effective software reliability through self-healing. **Computing and Informatics**, v. 29, n. 1, p. 93–115, 2010.

GURGEN, L. et al. Self-aware cyber-physical systems and applications in smart buildings and cities. In: **Proceedings of the** 16^{th} **Design, Automation and Test in Europe Conference and Exhibition (DATE)**. [S.l.: s.n.], 2013. p. 1149–1154.

HAKIRI, A.; GOKHALE, A.; BERTHOU, P. Software-defined wireless mesh networking for reliable and real-time smart city cyber physical applications. In: **Proceedings of the** 27^{th} **International Conference on Real-Time Networks and Systems**. New York, USA: Association for Computing Machinery, 2019. p. 165–175.

HAMDAOUI, B. et al. Iotshare: A blockchain-enabled iot resource sharing on-demand protocol for smart city situation-awareness applications. **IEEE Internet of Things Journal**, v. 7, n. 10, p. 10548–10561, 2020.

HAOUAS, H.; BOURCIER, J. Dependability-driven runtime management of service oriented architectures. In: **Proceedings of the** 4^{th} **International Workshop on Principles of Engineering Service-Oriented Systems (PESOS)**. New York, USA: IEEE, 2012. p. 15–21.

HARRISON, C. et al. Foundations for smarter cities. **IBM Journal of Research and Development**, v. 54, n. 4, p. 1–16, 2010.

HASEBE, K. et al. Traffic management for last-mile public transportation systems using autonomous vehicles. In: **Proceedings of the** 3^{th} **International Smart Cities Conference (ISC2)**. New York, USA: IEEE, 2017. p. 1–6.

HASEEB, K. et al. Trust management with fault-tolerant supervised routing for smart cities using internet of things. **IEEE Internet of Things Journal**, v. 9, n. 22, p. 22608–22617, 2022.

HE, Y. et al. Towards smarter cities: A self-healing resilient microgrid social network. In: **Proceedings of IEEE Power and Energy Society General Meeting (PESGM)**. New York, USA: IEEE, 2016. p. 1–5.

HENRY, M.; WOOD, G. Sensor validation: principles and standards. **ATP International**, v. 3, 04 2005.

HERNANDES, E. et al. Using gqm and tam to evaluate start - a tool that supports systematic review. **CLEI Electronic Journal**, v. 15, p. 3 – 3, 2012.

HOFBAUR, M. Model based reasoning for fault-tolerant robot hardware and software. In: **Proceedings of the** 16^{th} **International Workshop on Robotics in Alpe Adria Danube Region (RAAD)**. Berlin, Germany: Springer-Verlag, 2007.

HOLBERT, K. E.; HEGER, A. S.; ALANG-RASHID, N. K. Redundant sensor validation by using fuzzy logic. **Nuclear Science and Engineering**, Taylor & Francis, v. 118, n. 1, p. 54–64, 1994.

HORNING, J. J. et al. A program structure for error detection and recovery. In: **Symposium on Operating Systems**. Berlin, Germany: Springer-Verlag, 1974.

Hu, P.; Indulska, J.; Robinson, R. An autonomic context management system for pervasive computing. In: **Proceeding of the** 6^{th} **International Conference on Pervasive Computing and Communications (PerCom)**. New York, USA: IEEE, 2008. p. 213–223.

IBM. An architectural blueprint for autonomic computing. **IBM White Paper**, Citeseer, p. 34, 2006.

IBRAHIM, D. S.; MAHDI, A. F.; YAS, Q. M. Challenges and issues for wireless sensor networks: A survey. **J. Glob. Sci. Res**, v. 6, n. 1, p. 1079–1097, 2021.

IDWAN, S.; ZUBAIRI, J. A.; MAHMOOD, I. Smart solutions for smart cities: Using wireless sensor network for smart dumpster management. In: **2016 International Conference on Collaboration Technologies and Systems (CTS)**. [S.l.: s.n.], 2016. p. 493–497.

IEEE. **Standard Glossary of Software Engineering Terminology**. USA, 1990.

ISO/IEC/IEEE. **International Standard - Systems and software engineering–Vocabulary**. USA, 2017. 1-541 p.

JAIN, R.; SHAH, H. An anomaly detection in smart cities modeled as wireless sensor network. In: **2016 International Conference on Signal and Information Processing (IConSIP)**. [S.l.: s.n.], 2016. p. 1–5.

JAVADZADEH, G.; RAHMANI, A. M. Fog computing applications in smart cities: A systematic survey. **Wireless Networks**, v. 26, n. 2, p. 1433–1457, Feb 2020.

JESUS, T. C. et al. A dependability-aware approach for dynamic mobile sink repositioning in smart cities applications. In: **2022 IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2022. p. 1–7.

KASINATHAN, M. et al. An artificial neural network approach for the discordance sensor data validation for scram parameters. In: **Proceedings of the** 1^{th} **International Conference on Advancements in Nuclear Instrumentation, Measurement Methods and their Applications**. [S.l.: s.n.], 2009.

KAUR, M. J.; MAHESHWARI, P. Building smart cities applications using iot and cloud-based architectures. In: **2016 International Conference on Industrial Informatics and Computer Systems (CIICS)**. [S.l.: s.n.], 2016. p. 1–5.

KEPHART, J.; CHESS, D. The vision of autonomic computing. **IEEE Computer**, v. 36, n. 1, p. 41–50, 2003.

KHAN, H. et al. Privacy preserving data aggregation with fault tolerance in fog-enabled smart grids. **Sustainable Cities and Society**, v. 64, p. 102522, 01 2021.

KHAN, M. A.; KIM, H.-c.; PARK, H. Leveraging machine learning for fault-tolerant air pollutants monitoring for a smart city design. **Electronics**, v. 11, n. 19, p. 3122, 2022.

KHANG, A. et al. **Smart Cities: IoT Technologies, big data solutions, cloud platforms, and cybersecurity techniques**. [S.l.]: CRC Press, 2023.

KHARCHENKO, V. et al. UAV fleet as a dependable service for smart cities: Model-based assessment and application. **Smart Cities**, v. 5, n. 3, p. 1151–1178, 2022.

KIM, H.; BEN-OTHMAN, J. Famu: Fault-tolerant mutual assisted virtual emotion barrier system using intelligent smart UAVs. **IEEE Transactions on Vehicular Technology**, v. 70, n. 10, p. 10845–10852, 2021.

KIM, Y.; KIM, E.-k. Autonomic service reconfiguration in a ubiquitous computing environment. In: **Proceedings of the** ^{4}th **Parallel and Distributed Processing and Applications (ISPA)**. Berlin, Germany: Springer-Verlag, 2006. p. 584–593.

KITCHENHAM, B. **Procedures for performing systematic reviews**. [S.l.], 2004.

KLÖS, V.; GÖTHEL, T.; GLESNER, S. Comprehensible and dependable self-learning self-adaptive systems. **Journal of Systems Architecture**, v. 85-86, p. 28 – 42, 2018.

Kooli, M.; Di Natale, G. A survey on simulation-based fault injection tools for complex systems. In: **Proceedings of the 9th IEEE International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)**. Santorini, Greece: IEEE, 2014. p. 1–6.

KRUPITZER, C. et al. A survey on engineering approaches for self-adaptive systems. **Pervasive and Mobile Computing**, v. 17, p. 184 – 206, 2015. 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian.

KULKARNI, D.; TRIPATHI, A. A framework for programming robust context-aware applications. **IEEE Transactions on Software Engineering**, IEEE, New York, USA, v. 36, n. 2, p. 184–197, 2010.

KURESHI, R. R. et al. Use case of building an indoor air quality monitoring system. In: **2021 IEEE 7th World Forum on Internet of Things (WF-IoT)**. [S.l.: s.n.], 2021. p. 747–752.

LALANDA, P.; MCCANN, J. A.; DIACONESCU, A. **Autonomic Computing**. Berlin, Germany: Springer, 2013.

LAPRIE, J. C. C.; AVIZIENIS, A.; KOPETZ, H. (Ed.). **Dependability: Basic Concepts and Terminology**. Berlin, Germany: Springer-Verlag, 1992.

LAU, B. P. L. et al. Sensor fusion for public space utilization monitoring in a smart city. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 473–481, 2018.

LEE, P. A.; ANDERSON, T. **Fault Tolerance: Principles and Practice**. 2nd. ed. Berlin, Germany: Springer-Verlag, 1990.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: **Eighth IEEE International Conference on Data Mining**. [S.l.: s.n.], 2008. p. 413–422.

LIU, L. et al. A fault-tolerant mobile sensing information gathering center (msigc) using public transport buses to instrument a smart city. In: **Proceedings of the** 9^{th} **International Conference on Advanced Infocomm Technology (ICAIT)**. New York, USA: IEEE, 2017. p. 233–238.

LU, H.; CHAN, W.; TSE, T. Testing context-aware middleware-centric programs: A data flow approach and an rfid-based experimentation. In: **Proceedings of the** 14^{th} **International Symposium on Foundations of Software Engineering**. New York, USA: Association for Computing Machinery, 2006. p. 242–252.

Lugo-Cordero, H. M.; Guha, R. K.; Ortiz-Rivera, E. I. An adaptive cognition system for smart grids with context awareness and fault tolerance. **IEEE Transactions on Smart Grid**, v. 5, n. 3, p. 1246–1253, 2014.

MANOGARAN, G.; NGUYEN, T. N. Displacement-aware service endowment scheme for improving intelligent transportation systems data exchange. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 11, p. 22467–22477, 2022.

McGee, E. T.; McGregor, J. D. Using dynamic adaptive systems in safety-critical domains. In: **Proceedings of the** 11^{th} **International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. New York, USA: IEEE, 2016. p. 115–121.

MIRZA, I. B.; GEORGAKOPOULOS, D.; YAVARI, A. Cyber-physical-social awareness platform for comprehensive situation awareness. **Sensors**, v. 23, n. 2, 2023.

MODARRESI, A.; SYMONS, J. Resilience and technological diversity in smart homes: A graph-theoretic approach to modeling iot systems with integrated heterogeneous networks. **Journal of Ambient Intelligence and Humanized Computing**, v. 11, 06 2020.

MODUKURI, K. et al. Autonomous middleware framework for sensor networks. In: **Proceedings of the** 2^{th} **International Conference on Pervasive Services (ICPS)**. New York, USA: IEEE, 2005. p. 17–26.

MOHAMED, N.; AL-JAROODI, J.; JAWHAR, I. Towards fault tolerant fog computing for iot-based smart city applications. In: **Proceedings of the** 9^{th} **Annual Computing and Communication Workshop and Conference (CCWC)**. New York, USA: IEEE, 2019. p. 0752–0757.

MOHAPATRA, H.; RATH, A. K. A fault tolerant routing scheme for advanced metering infrastructure: an approach towards smart grid. **Cluster Computing**, v. 24, n. 3, p. 2193–2211, Sep 2021.

MORA-MORA, H. et al. A computational architecture based on rfid sensors for traceability in smart cities. **Sensors**, v. 15, n. 6, p. 13591–13626, 2015.

MOURAO, E. et al. Investigating the Use of a Hybrid Search Strategy for Systematic Reviews. In: **2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**. [S.l.: s.n.], 2017. p. 193–198.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. [S.l.]: Wiley, 2011.

Nair, R. R. et al. Fault-tolerant formation control of nonholonomic robots using fast adaptive gain nonsingular terminal sliding mode control. **IEEE Systems Journal**, v. 13, n. 1, p. 1006–1017, 2019.

NANDURY, S. V.; BEGUM, B. A. Smart WSN-based ubiquitous architecture for smart cities. In: **2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2015. p. 2366–2373.

NASCIMENTO, L. Vianna do; OLIVEIRA, J. Palazzo Moreira de. Decentralized, distributed and fault-tolerant context recognition architectures for smart cities: A systematic mapping. **Reviews on Computer Science (ROCS)**, v. 1, n. 1, 2021.

Ning-jiang Chen; Pan Lin. An adaptive fault-tolerant mechanism for web services based on context awareness. In: **Proceedings of the** 4^{th} **International Conference on Computer Science Education (ICCSE)**. New York, USA: IEEE, 2009. p. 898–903.

OIKONOMOU, G. et al. The contiki-ng open source operating system for next generation iot devices. **SoftwareX**, Elsevier, v. 18, p. 101089, 2022.

Oreizy et al., P. An architecture-based approach to self-adaptive software. **IEEE Intelligent Systems**, v. 14, n. 3, p. 54–62, 1999.

PASTóRIO, A. F.; CAMARGO, E. T. de. Geolocation techniques in lorawan networks as a fault tolerance approach in GPS-based tracking devices. In: **2021 Third South American Colloquium on Visible Light Communications (SACVLC)**. [S.l.: s.n.], 2021. p. 01–06.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. **Information and Software Technology**, v. 64, p. 1 – 18, 2015.

PIRES, I. et al. Validation techniques for sensor data in mobile health applications. **Journal of Sensors**, v. 2016, 01 2016.

POWAR, V. et al. Sensor networks for hydrometric monitoring of urban watercourses. In: **Proceedings of the** 16^{th} **International Conference on Smart Cities: Improving Quality of Life Using ICT IoT and AI (HONET-ICT)**. New York, USA: IEEE, 2019. p. 085–089.

PRADHAN, S. et al. Achieving resilience in distributed software systems via self-reconfiguration. **Journal of Systems and Software**, v. 122, p. 344–363, 2016. ISSN 0164-1212.

PRIVAT, G.; ZHAO, M.; LEMKE, L. Towards a shared software infrastructure for smart homes, smart buildings and smart cities. In: **Proceedings of the International Workshop on Emerging Trends in the Engineering of CyberPhysical Systems (EITEC)**. [S.l.: s.n.], 2014.

PUIU, D. et al. Citypulse: Large scale data analytics framework for smart cities. **IEEE Access**, v. 4, p. 1086–1108, 2016.

PUTRA, H. Y.; PUTRA, H.; KURNIAWAN, N. B. Big data analytics algorithm, data type and tools in smart city: A systematic literature review. In: **2018 International Conference on Information Technology Systems and Innovation (ICITSI)**. [S.l.: s.n.], 2018. p. 474–478.

Ramakrishnan, S. On self-adaptive process-based dependable web service composition. In: **Proceedings of Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns (COMPUTATIONWORLD)**. New York, USA: IEEE, 2009. p. 173–179.

Rocha, L. S.; Andrade, R. M. C. Towards a formal model to reason about context-aware exception handling. In: **Proceedings of the** 5^{th} **International Workshop on Exception Handling (WEH)**. New York, USA: IEEE, 2012. p. 27–33.

ROCHA, N. et al. Systematic literature review of context-awareness applications supported by smart cities' infrastructures. **SN Applied Sciences**, v. 4, 2022.

RUBí, J. N. S.; GONDIM, P. R. de L. Iot-based platform for environment data sharing in smart cities. **International Journal of Communication Systems**, v. 34, n. 2, p. e4515, 2021.

SAID, A. M.; KAMAL, A. E.; AFIFI, H. An intelligent parking sharing system for green and smart cities based iot. **Computer Communications**, v. 172, p. 10–18, 2021.

SÃNCHEZ-CORCUERA, R. et al. Smart cities survey: Technologies, application domains and challenges for the cities of the future. **International Journal of Distributed Sensor Networks**, v. 15, n. 6, June 2019.

SANTANA, E. F. Z. et al. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 50, n. 6, 2017.

SASU, L.; PUIU, D.; NECHIFOR, S. Fault recovery mechanism for smart city environments. In: **Proceedings of the** 20^{th} **Jubilee International Conference on Intelligent Engineering Systems (INES)**. New York, USA: IEEE, 2016. p. 57–62.

SCHöLKOPF, B. et al. Estimating the support of a high-dimensional distribution. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 13, n. 7, p. 1443–1471, 2001. ISSN 0899-7667.

SHAH, S. et al. Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers? **IEEE Access**, v. 7, p. 91885 – 91903, 2019.

SHAHROUR, I.; XIE, X. Role of internet of things (iot) and crowdsourcing in smart city projects. **Smart Cities**, v. 4, n. 4, p. 1276–1292, 2021.

SHAMSI, J. A. Resilience in smart city applications: Faults, failures, and solutions. **IT Professional**, v. 22, n. 6, p. 74–81, 2020.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, [Oxford University Press, Biometrika Trust], v. 52, n. 3/4, p. 591–611, 1965.

SHARIFI, A. A typology of smart city assessment tools and indicator sets. **Sustainable Cities and Society**, v. 53, p. 101936, 2020.

SHAW, M. Beyond objects: A software design paradigm based on process control. **ACM Software Engineering Notes**, v. 20, p. 20–1, 1995.

SHI, E. An improved real-time adaptive kalman filter for low-cost integrated gps/ins navigation. In: **Proceedings of 2012 International Conference on Measurement, Information and Control**. [S.l.: s.n.], 2012. v. 2, p. 1093–1098.

SIQUEIRA, B. R. et al. Testing of adaptive and context-aware systems: approaches and challenges. **Software Testing, Verification and Reliability**, v. 31, n. 7, p. e1772, 2021.

Sliem, M.; Salmi, N.; Ioualalen, M. Towards reliability and performance prediction of autonomic systems with self-healing and protection. In: **Proceedings of the** 4^{th} **International Conference on Cloud and Autonomic Computing (ICCAC)**. New York, USA: IEEE, 2014. p. 35–43.

SOHRABY, K.; MINOLI, D.; ZNATI, T. Applications of wireless sensor networks. In: _____. **Wireless Sensor Networks: Technology, Protocols, and Applications**. [S.l.: s.n.], 2007. p. 38–74.

SOUZA, K. **Artifacts of the Research Process**. Zenodo, 2024. Disponível em: <https://zenodo.org/records/12752460>.

SOUZA, K. E.; FERRARI, F. C. A Systematic Review of Fault Tolerance Techniques for Adaptive and Context-aware Systems. In: **Proceedings of the 3^{rd} IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)**. [S.l.]: IEEE, 2022. p. 21–30.

STEINBAUER, G.; MÖRTH, M.; WOTAWA, F. Real-time diagnosis and repair of faults of robot control software. In: **RoboCup 2005: Robot Soccer World Cup IX**. Berlin, Germany: Springer-Verlag, 2006. p. 13–23.

SYED, A. S. et al. IoT in Smart Cities: A Survey of Technologies, Practices and Challenges. **Smart Cities**, v. 4, n. 2, p. 429–475, 2021.

TALEBKHAH, M. et al. Iot and big data applications in smart cities: Recent advances, challenges, and critical issues. **IEEE Access**, v. 9, p. 55465–55484, 2021.

TANCEV, G.; TORO, F. G. Sequential recalibration of wireless sensor networks with (stochastic) gradient descent and mobile references. **Measurement: Sensors**, v. 18, p. 100115, 2021.

Tohma, Y. Incorporating fault tolerance into an autonomic-computing environment. **IEEE Distributed Systems Online**, v. 5, n. 2, p. 3/1–3/12, 2004.

United Nations. **World Urbanization Prospects: The 2018 Revision (ST/ESA/SER.A/420)**. New York, NY, USA, 2019.

V, S. et al. Random forest: a classification and regression tool for compound classification and QSAR modeling. **Journal of Chemical Information and Modeling**, 14632445, v. 10, 2003.

VARGAS-SANTIAGO, M. et al. Autonomic web services enhanced by asynchronous checkpointing. **IEEE Access**, v. 6, p. 5538–5547, 2018.

WAN, J. et al. M2m communications for smart city: An event-based architecture. In: **Proceedings of the 12^{th} International Conference on Computer and Information Technology (ICCSIT)**. [S.l.: s.n.], 2012. p. 895–900.

WANG, H.; CHAN, W.; TSE, T. Improving the effectiveness of testing pervasive software via context diversity. **ACM Transactions on Autonomous and Adaptive Systems**, New York/NY - USA, v. 9, n. 2, p. 1–28, 2014.

WANG, X.; LIU, Y.; CHOO, K.-K. R. Fault-tolerant multisubset aggregation scheme for smart grid. **IEEE Transactions on Industrial Informatics**, v. 17, n. 6, p. 4065–4072, 2021.

WASHBURN, D.; SINDHU, U. Helping cios understand 'smart city' initiatives. **Growth**, 2009.

Weyns et al., D. On patterns for decentralized control in self-adaptive systems. In: **Proceedings of Software Engineering Workshop for Self-Adaptive Systems II**. Berlin, Germany: Springer-Verlag, 2013. v. 7475 LNCS, p. 76–107.

WIERINGA, R. et al. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. **REQUIREMENTS ENGINEERING**, 11, n. 1, p. 102–107, 2006.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: **Proceedings of the** 18^{th} **International Conference on Evaluation and Assessment in Software Engineering (EASE)**. New York, USA: Association for Computing Machinery, 2014. p. 1–10.

WOHLIN, C. et al. **Experimentation in Software Engineering**. Berlin, Heidelberg: Springer, 2012. ISBN 978-3-642-29044-2.

XU, C.; CHEUNG, S. C. Inconsistency detection and resolution for context-aware middleware support. In: **Proceedings of the** 10^{th} **European Software Engineering Conference Held Jointly with** 13^{th} **ACM SIGSOFT International Symposium on Foundations of Software Engineering**. New York, USA: Association for Computing Machinery, 2005. p. 336—-345.

Xu, C. et al. Environment rematching: Toward dependability improvement for self-adaptive applications. In: **Proceedings of the** 27^{th} **International Conference on Automated Software Engineering (ASE)**. New York, USA: IEEE, 2013. p. 592–597.

XU, J. et al. Fault tolerance in concurrent object-oriented software through coordinated error recovery. In: **Proceedings of the** 25^{th} **International Symposium on Fault-Tolerant Computing (FTCS)**. New York, USA: IEEE, 1995. p. 499–508.

YANG, T.; PARK, S.; KIM, S.-H. Collaborative reliable event transport based on mobile-assisted sensing in urban digital twin. **Electronics**, v. 11, n. 10, p. 1550, 2022.

YIN, C. et al. A literature survey on smart cities. **Science China Information Sciences**, v. 58, 08 2015.

Yoon, T. et al. A formal modeling for exceptions in context-aware systems. In: **Prceedings of the** 38^{th} **International Computer Software and Applications Conference (COMPSAC) Workshops**. New York, USA: Association for Computing Machinery, 2014. p. 734–739.

YU, Z. et al. Distributed adaptive fault-tolerant time-varying formation control of unmanned airships with limited communication ranges against input saturation for smart city observation. **IEEE Transactions on Neural Networks and Learning Systems**, v. 33, n. 5, p. 1891–1904, 2022.

_____. Distributed fractional-order intelligent adaptive fault-tolerant formation-containment control of two-layer networked unmanned airships for safe observation of a smart city. **IEEE Transactions on Cybernetics**, v. 52, n. 9, p. 9132–9144, 2022.

YUAN, W. et al. An alternative reliability method to evaluate the regional traffic congestion from GPS data obtained from floating cars. **IET Smart Cities**, v. 3, n. 2, p. 79–90, 2021.

ZAITER, M.; HACINI, S. Towards exploring context to insure fault tolerance in home automation medical system. In: **Proceedings of the 5^{th} International Congress on Information Science and Technology (CiSt)**. New York, USA: IEEE, 2018. p. 29–35.

ZANELLA, A. et al. Internet of things for smart cities. **IEEE Internet of Things Journal**, v. 1, n. 1, p. 22–32, 2014.

ZHANG, D. et al. ibat: detecting anomalous taxi trajectories from gps traces. In: **Proceedings of the 13th International Conference on Ubiquitous Computing**. New York, NY, USA: Association for Computing Machinery, 2011. p. 99–108.

ZHOU, P. et al. A comprehensive technological survey on the dependable self-management CPS: from self-adaptive architecture to self-management strategies. **Sensors**, MDPI AG, v. 19, n. 5, p. 1033, 2019.

Zhou, X. et al. A Map of Threats to Validity of Systematic Literature Reviews in Software Engineering. In: **Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC)**. [S.l.: s.n.], 2016. p. 153–160.