Programa de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Cristiano Welter

Model of Things: Uma Abordagem de Desenvolvimento de Software Dirigida por Modelos para Aplicações Cloud of Things

São Leopoldo, 2019

Cristiano Welter

MODEL OF THINGS: UMA ABORDAGEM DE DESENVOLVIMENTO DE SOFTWARE DIRIGIDA POR MODELOS PARA APLICAÇÕES CLOUD OF THINGS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Kleinner Silva Farias de Oliveira

São Leopoldo
2019

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Welter, Cristiano

Model of Things: Uma Abordagem de Desenvolvimento de Software Dirigida por Modelos para Aplicações Cloud of Things / Cristiano Welter — 2019.

107 f.: il.; 30 cm.

Dissertação (mestre) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2019.

"Orientador: Prof. Dr. Kleinner Silva Farias de Oliveira, Unidade Acadêmica de Pesquisa e Pós-Graduação".

1. Desenvolvimento Dirigido por Modelos. 2. Internet das Coisas. 3. Computação em Nuvem. 4. Nuvem das Coisas. I. Título.

CDU 004

Bibliotecária responsável: Amanda Schuster — CRB 10/2517

Cristiano Welter

Model of Things: Uma Abordagem de Desenvolvimento de Software Dirigida por Modelos para Aplicações Cloud of Things

Dissertação apresentada à Universidade do Vale do Rio dos Sinos – Unisinos, como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Aprovado em 23 de abril de 2019

BANCA EXAMINADORA

Prof. Dr. Kleinner Silva Farias de Oliveira – PPGCA/UNISINOS

Prof. Dr. Jorge Luis Victória Barbosa – PPGCA/UNISINOS

Prof. Dr. Gustavo Pessin – Instituto Tecnológico Vale

Prof. Dr. Kleinner Silva Farias de Oliveira (Orientador)

Visto e permitida a impressão
São Leopoldo

Prof. Dr. Rodrigo da Rosa Righi
Coordenador PPG em Computação Aplicada

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001.

AGRADECIMENTOS

Primeiramente quero agradecer a todos os professores que fizeram parte do meu percurso universitário, que de alguma forma contribuíram para que eu tenha alcançado esse objetivo. Mas em especial gostaria de agradecer ao Professor Doutor Kleinner Silva Farias de Oliveira por todo o acompanhamento ao longo desta pesquisa, pela sua disponibilidade para prontamente resolver as dúvidas que surgiram e principalmente pelo apoio motivacional que proporcionou no decorrer do trabalho.

O mais importante dos agradecimentos vai para minha noiva Caroline, quem me motivou a fazer este mestrado e que esteve ao meu lado durante estes dois anos com palavras de incentivo e principalmente muita paciência pelos momentos que precisei estar distante. Obrigado de todo o coração, te amo.

Também gostaria de agradecer aos meus familiares e amigos por todo o apoio, paciência e conselhos durante o meu percurso escolar. Esta dissertação não seria concretizável sem a ajuda de todos eles.

Por último agradeço ao Programa de Pós-Graduação em Computação Aplicada da Unisinos pela oportunidade de realizar esse mestrado e também a CAPES pelo apoio financeiro a esta pesquisa.

“A verdadeira motivação vem de realização, desenvolvimento pessoal, satisfação no trabalho e reconhecimento.”
(Frederick Herzberg)

RESUMO

A indústria da informação deu grandes passos e está ganhando reconhecimento com a integração da *Internet* das Coisas (IoT). Porém, o uso efetivo do IoT é muito mais do que apenas conectar coisas, abrange como uma preocupação principal a gestão e a transformação dos dados gerados em ideias e benefícios empresariais. A ampla adoção da IoT tem levado à produção de sistemas complexos, os quais produzem uma grande quantidade de dados que interagem com outros sistemas e/ou serviços. Assim, existe a necessidade de um ambiente forte e flexível para o suporte a essas aplicações. A Computação em Nuvem fornece uma base sólida para o compartilhamento de recursos de forma flexível, e quando utilizada em conjunto com aplicações de IoT para integração dos dados, surge um novo paradigma chamado de Nuvem das Coisas (CoT). No entanto, o desenvolvimento dos sistemas CoT apresenta um conjunto de desafios, incluindo a complexidade dos dispositivos de IoT e o gerenciamento dos dados heterogêneos, bem como a escalabilidade do sistema. Além disso, exige dos desenvolvedores um conjunto diversificado de habilidades e conhecimentos que abrangem o domínio do problema, o processamento do sinal dos sensores, algoritmos, infraestruturas de implantação, entre outros. Nesse contexto, as abordagens de Desenvolvimento Dirigido por Modelos (MDD) apresentam um potencial relevante para lidar de forma adequada com esses problemas pelo fato de oferecerem um nível elevado de abstração o que reduz a complexidade dos artefatos de *software* e os esforços necessários para produzi-los. Tal característica proporciona reusabilidade, portabilidade e interoperabilidade, aumentando a produtividade no processo e a qualidade do *software*. Portanto, este trabalho propõe o MoT, uma abordagem de desenvolvimento MDD para aplicações de CoT. Através de transformações de modelos para modelos e modelos para código, a abordagem automatiza a geração da uma aplicação para conectar dispositivos de IoT em uma infraestrutura de computação em nuvem. O MoT foi validado através de um estudo de caso com cenários reais da IoT, e posteriormente o protótipo foi avaliado por usuários através do modelo de aceitação de tecnologia (TAM). Essas avaliações mostraram que o MoT trata-se de uma abordagem promissora ao permitir a criação de aplicações para CoT através da abstração dos detalhes técnicos das tecnologias, e da heterogeneidade dos provedores de computação em nuvem.

Palavras-chave: Desenvolvimento Dirigido por Modelos. Internet das Coisas. Computação em Nuvem. Nuvem das Coisas.

ABSTRACT

The information industry has made great strides and is gaining recognition with the integration of the Internet of Things (IoT). However, the effective use of IoT is much more than just connecting things, it covers as a main concern the management and use of the generated data into business opportunities. The widespread adoption of IoT has led to the production of complex systems, which produce a large amount of data that interacts with other systems and/or services. Thus, there is a need for a strong and flexible environment to support these applications. Cloud Computing provides a solid foundation for resource sharing in a flexible way, and when used in conjunction with IoT applications for data integration, a new paradigm called the Cloud of Things (CoT) emerges. However, the development of CoT systems presents a number of challenges, including the complexity of IoT devices and the management of heterogeneous data, as well as the scalability of the system. In addition, it requires developers to have a diverse set of skills and knowledge that encompasses problem mastery, sensor signal processing, algorithms, deployment infrastructures, and more. In this context, Model Driven Development (MDD) approaches present a relevant potential to deal adequately with these problems by offering a high level of abstraction which reduces the complexity of software artifacts and efforts necessary to produce them. Such a feature provides reusability, portability and interoperability, increasing process productivity and software quality. Therefore, this work proposes MoT, an MDD development approach for CoT applications. Through model transformations for models and models for code, the approach automates the generation of an application to connect IoT devices to a cloud computing infrastructure. The MoT was validated through a case study with real IoT scenarios, and later the prototype was evaluated by users through the technology acceptance model (TAM). These evaluations have shown that MoT is a promising approach by enabling the creation of CoT applications by abstracting the technical details of technologies and the heterogeneity of cloud computing providers.

Keywords: Model Driven Development. Internet of Things. Cloud Computing. Cloud of Things.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

CIM *Computation Independent Model* / Modelo Independente de Computação

CoT *Cloud of Things* / Nuvem das Coisas

DSL *Domain-Specific Language* / Linguagem Específica de Domínio

IoT *Internet of Things / Internet* das Coisas

MDA *Model-Driven Architecture* / Arquitetura Dirigida a Modelos

MDD *Model-Driven Development* / Desenvolvimento Dirigido a Modelos

MDE *Model-Driven Engineering* / Engenharia Dirigido a Modelos

MOF *Meta-Object Facility* / Infraestrutura de MetaObjetos

OMG *Object Management Group* / Grupo de Gerenciamento de Objetos

PIM *Platform Independent Model* / Modelo Independente de Plataforma

PSM *Platform Specific Model* / Modelo Específico de Plataforma

TAM *Technology Acceptance Model* / Modelo de Aceitação de Tecnologia

UML *Unified Modeling Language* / Linguagem de Modelagem Unificada

XMI *XML Metadata Interchange* / Intercâmbio de Metadados em XML

SUMÁRIO

1 INTRODUÇÃO

A indústria da informação está ganhando reconhecimento com a integração de objetos inteligentes que são capazes de perceber o mundo físico através do uso de sensores, afetar o mundo físico executando ações através do uso de atuadores, engajar usuários interagindo com eles sempre que necessário, processar os dados coletados e comunicar esses dados a redes e sistemas externos. A integração desses objetos na *Internet* deu origem ao conceito de *Internet* das Coisas (do inglês, *Internet Of Things* - IoT).

Recentes avanços tecnológicos têm promovido um enorme crescimento no número desses objetos inteligentes disponíveis para uso. A Gartner, Inc. (GARTNER, 2016) diz que até 2020, 21 bilhões de dispositivos de IoT estarão em uso em todo o mundo. Estes dispositivos estarão presentes nos mais diversos objetos que vão desde roupas, veículos, eletrodomésticos, e irão produzir um impacto significativo no dia a dia dos seres humanos com o surgimento de novos sistemas em vários domínios de aplicação, como telecomunicações, medicina, monitoramento ambiental, agricultura de precisão, rastreabilidade de alimentos, edifícios inteligentes, segurança e privacidade, monitoramento urbano e muito mais.

Devido ao grande número de dispositivos conectados, que são desenvolvidos por diversos fabricantes, a principal característica dos sistemas IoT é a heterogeneidade e complexidade entre seus componentes, ou seja, grande diversidade de tecnologias de *hardware* e *software* envolvidas no ambiente.

Essa ampla adoção de IoT levará à produção de sistemas complexos, os quais produzem uma grande quantidade de dados, quantidade esta sem precedentes na história da humanidade (GUBBI et al., 2013). Estes dados deverão ser armazenados, processados e analisados a fim de que sejam geradas informações úteis para atender aos requisitos propostos.

Assim, existe a necessidade de um ambiente flexível para ajustar a quantidade de recursos computacionais consumidos de acordo com a necessidade da aplicação. Nesse contexto a computação em nuvem (do inglês, *cloud computing*) complementa o conceito de IoT, no sentido de prover escalabilidade e serviços sob demanda (CUBO; NIETO; PIMENTEL, 2014) (CAVALCANTE et al., 2016).

A computação em nuvem é um modelo que permite o acesso a uma rede compartilhada, conveniente e *on-demand*, de um *pool* de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e lançados com o mínimo esforço de gerenciamento ou interação do provedor de serviços (AAZAM et al., 2016).

A integração entre IoT e computação em nuvem dá origem ao paradigma chamado de Nuvem das Coisas (do inglês, *Cloud of Things* - CoT) (ZHOU et al., 2013). Essa integração é necessária a fim de possibilitar, processar e armazenar uma grande quantidade de dados e fornecê-los para que aplicações sejam construídas, com requisitos de disponibilidade, capacidade de processamento e alocação de recursos sob demanda.

No entanto, o desenvolvimento de aplicações CoT é um desafio, pois envolve lidar com uma ampla gama de questões relacionadas aos dispositivos de IoT e questões de infraestrutura e serviços de nuvem. Exige dos desenvolvedores um conjunto diversificado de habilidades e conhecimentos abrangendo o domínio, incorporar *hardware*, *software* e rede de computadores de forma inteligente, infraestrutura de implantação e muito mais. Nesse sentido, abordagens de Desenvolvimento Dirigidas por Modelos (do inglês, Model Driven Development - MDD), apresentam um potencial relevante para contribuir na melhoria da qualidade do processo e do produto de *software*.

Abordagens dirigidas por modelos já são amplamente utilizadas na engenharia de *software* para reduzir a complexidade do desenvolvimento de aplicações, pois permitem especificar sistemas em um nível mais alto de abstração, utilizando conceitos mais próximos do domínio do problema e reduzindo os detalhes da tecnologia da plataforma de implementação. E a partir da especificação transformações automáticas poderão ser aplicadas para gerar código para plataformas específicas, reduzindo o esforço de desenvolvimento.

1.1 Problemática

Conforme mencionado anteriormente, o desenvolvimento de aplicações CoT ainda apresenta grandes desafios, principalmente envolvendo dificuldades relacionadas aos conceitos de IoT e detalhes técnicos de infraestrutura dos provedores de computação em nuvem. Neste sentido, listam-se a seguir as problemáticas que são exploradas neste trabalho:

- **Complexidade:** Embora aplicações de IoT e computação em nuvem tenham sido amplamente aceitas na indústria, desenvolver aplicações CoT ainda é uma tarefa complexa e propensa a erros. Isso pode ser explicado por duas razões. Primeira, o desenvolvimento de aplicações CoT exige a formação de times multidisciplinares, com conhecimento em diferentes tipos de linguagens, arquiteturas, protocolos de comunicação, entre outros temas (PATEL; CASSOU, 2015). Os times de desenvolvimento devem possuir desde especialistas em domínio até especialistas em dispositivos móveis, sensores, redes e plataformas de nuvem (JAISWAL et al., 2015). Porém, times de desenvolvimento com esta qualificação são caros e difíceis de serem formados. A segunda razão seria que as abordagens atuais de desenvolvimento de aplicações CoT exigem que os desenvolvedores trabalhem com código em baixo nível de abstração, exigindo conhecimento aprofundado de tecnologias específicas, satisfazendo geralmente apenas as necessidades de desenvolvedores experientes. Logo, desenvolver aplicações CoT torna-se um desafio e uma atividade altamente propensa a erros, principalmente para aqueles desenvolvedores inexperientes que precisam superar uma curva de aprendizagem alta (PRAMUDIANTO et al., 2016).

- **Heterogeneidade de ambientes de nuvem e IoT:** Um grande desafio para CoT está relacionado à ampla heterogeneidade de dispositivos, sistemas operacionais, plataformas e

serviços disponíveis. Por um lado, é preciso lidar com dispositivos de vários fabricantes e com diferentes funcionalidades e protocolos de rede, um problema que cria barreiras operacionais para usar esses dispositivos de uma forma abrangente e integrada (DELICATO; PIRES; BATISTA, 2013). Por outro lado, a heterogeneidade e a falta de padronização na computação em nuvem dificultam o uso de serviços oferecidos por vários provedores. Esse problema afeta diretamente o desenvolvimento e a implantação de aplicativos, uma vez que eles se tornam altamente acoplados aos serviços e restrições de um único provedor de nuvem (JOSEP et al., 2010). Portanto, mais pesquisas são necessárias para fornecer plataformas capazes de lidar com a heterogeneidade inerente dos ambientes de nuvem e IoT, enquanto alavancam o desenvolvimento e a implantação de aplicações que se beneficiam da integração de uma infinidade de dispositivos de IoT e podem usar serviços de nuvem oferecidos por diferentes provedores. Além disso, é importante fornecer aos desenvolvedores mecanismos flexíveis que permitam mudar facilmente de um determinado provedor de serviços para outro, bem como selecionar os serviços em nuvem que atendem aos requisitos da aplicação (CAVALCANTE et al., 2016).

1.2 Questões de Pesquisa

Considerando hoje o grande número de dispositivos conectados, a previsão de crescimento desses números para os próximos anos e os problemas apresentados anteriormente, fica evidente a necessidade de propor melhorias nas abordagens de desenvolvimento desse tipo de aplicações, enfatizando que essas beneficiam-se das vantagens oferecidas pelos serviços de computação em nuvem. Nesse contexto as abordagens de desenvolvimento dirigidas por modelo apresentam um potencial relevante para contribuir na melhoria do processo de desenvolvimento e na qualidade do *software*.

Os trabalhos investigados na literatura que abordam esse tema representam um avanço não só para a academia, mas também para a indústria, entretanto ainda existem limitações nas abordagens utilizadas. Sendo assim, este trabalho busca investigar duas lacunas que são formuladas através de duas Questões de Pesquisa (QP) a seguir:

- **QP-1:** Com o uso de uma abordagem MDD é possível construir uma aplicação de IoT nativa da nuvem, com pouco ou nenhum esforço de desenvolvimento manual e configuração de serviços?

- **QP-2:** Através da abstração dos componentes de IoT e dos serviços de computação em nuvem, usuários com pouca ou nenhuma experiência de desenvolvimento conseguiriam criar suas próprias aplicações de CoT?

O desenvolvimento desta pesquisa buscará responder essas questões. Para auxiliar neste estudo foram definidos alguns objetivos, que são apresentados na próxima seção.

1.3 Objetivos

Após contextualizar este estudo, apurar os principais problemas pesquisados, e abordar algumas das limitações dos trabalhos investigados, estabeleceu-se o objetivo principal deste trabalho, podendo a partir deste, elaborar as ações necessárias para o seu cumprimento.

Objetivo principal: Propor uma abordagem para auxiliar o desenvolvimento de aplicações CoT, reduzindo a complexidade da integração dos dispositivos de IoT e agnóstica dos provedores de computação em nuvem, através de uma abordagem dirigida por modelos.

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Realizar um estudo dos conceitos fundamentais referentes as abordagens do desenvolvimento dirigidas por modelos, *Internet* das Coisas e computação em nuvem para compor o embasamento teórico desta pesquisa;
- Obter um panorama atual sobre o estado da arte, procurando entender como as abordagens dirigidas por modelos vem sendo aplicadas no desenvolvimento de *software*, principalmente em aplicações CoT;
- Propor uma abordagem baseada em modelos para melhorar o desenvolvimento de aplicações CoT através da abstração dos componentes de IoT e detalhes de implantação das plataformas de computação em nuvem;
- Implementar um protótipo da ferramenta para dar suporte a abordagem proposta; e
- Realizar estudos empíricos para avaliar e mensurar a efetividade da abordagem proposta, tendo em vista produzir conhecimento sobre sua aplicação.

A seção seguinte detalha a metodologia utilizada no desenvolvimento deste trabalho.

1.4 Metodologia

Este trabalho é classificado quanto a sua natureza como uma pesquisa aplicada, pelo fato que existe a implementação de um protótipo da ferramenta baseada na abordagem proposta que possibilitará demonstrar o seu uso na prática (PRODANOV; FREITAS, 2013).

É um trabalho com objetivos exploratórios, que busca melhorar o processo de desenvolvimento de aplicações CoT a partir de uma abordagem dirigida por modelos. O desenvolvimento da pesquisa envolve buscas de informações sobre o tema, através de levantamento bibliográfico e análise de exemplos (PRODANOV; FREITAS, 2013).

Diversos procedimentos técnicos são utilizados no desenvolvimento deste trabalho, os quais caracterizam esta pesquisa segundo PRODANOV; FREITAS (2013) como:

- Bibliográfica: se utiliza da materiais já publicados sobre o tema da pesquisa;

- Experimental: a solução proposta envolve o desenvolvimento de um protótipo de *software* que será submetido a testes e avaliações;

- Levantamento (*survey*): faz uso de questionários aplicados aos participantes, para coleta de dados que serão analisados na avaliação da abordagem;

- Estudo de caso: busca realizar um estudo minucioso e profundo da abordagem em situações de uso real.

Por fim, a pesquisa é classificada como qualitativa quanto ao método, uma vez que as percepções de uso do protótipo serão classificadas e analisadas, buscando avaliar a aceitação da abordagem proposta a fim de atender as questões de pesquisa apresentadas na Seção 1.2.

A metodologia de trabalho a ser utilizado no desenvolvimento desta pesquisa é formado por quatro etapas, que são descritas a seguir:

- **Etapa 1:** A primeira etapa consiste na realização de uma revisão da literatura sobre como as abordagens de desenvolvimento dirigidas por modelos vem sendo aplicados no contexto de aplicações de IoT e/ou computação em nuvem. Essa revisão tem por objetivo analisar as técnicas disponíveis na academia, bem como identificar suas limitações com o intuito de elencar novas oportunidades de pesquisas. A revisão da literatura é usada para reunir informações sobre determinados tópicos, uma vez que procura fornecer maior familiaridade com o tema, além de aprofundar seus conceitos preliminares (MARCONI; LAKATOS, 2003).

- **Etapa 2:** A segunda etapa consiste na elaboração de uma abordagem dirigida por modelos para aplicações de CoT, que contemple as lacunas identificadas nos trabalhos encontrados na literatura.

- **Etapa 3:** A terceira etapa consiste no desenvolvimento de um protótipo para dar suporte a abordagem proposta. Nesta etapa são apresentados os aspectos de implementação da abordagem e suas principais características.

- **Etapa 4:** Por fim, a última etapa compreende a realização de experimentos no desenvolvimento de aplicações para CoT seguindo a abordagem proposta com o apoio da ferramenta desenvolvida, visando obter pareceres sobre as contribuições e benefícios do uso da abordagem proposta.

1.5 Organização do Trabalho

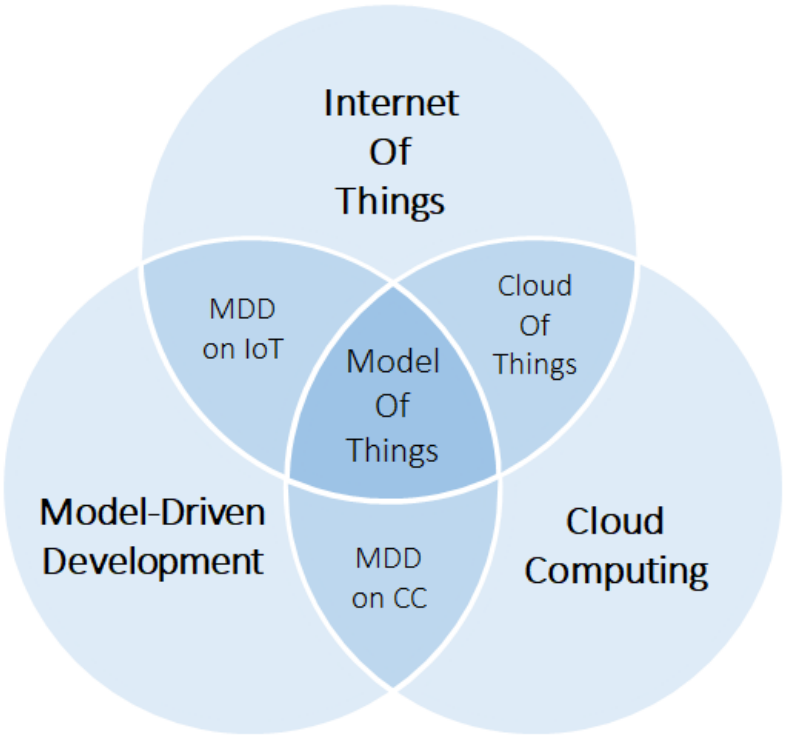
Este trabalho está organizado em sete capítulos, conforme descrito a seguir: O Capítulo 2 descreve os conceitos fundamentais para entendimento desta pesquisa. O Capítulo 3 apresenta uma análise comparativa entre trabalhos encontrados na literatura e descreve as oportunidades

de pesquisa. O Capítulo 4 além de apresentar a abordagem proposta, detalha os processos realizados para a transformação de modelos, as principais características, a arquitetura, e os aspectos de implementação. O Capítulo 5 descreve o método utilizado para avaliar a abordagem e discute os resultados encontrados. Por fim, o Capítulo 6 apresenta as considerações finais deste trabalho, como também as contribuições, as limitações e sugestões de trabalhos futuros para a continuação desta pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos fundamentais para o entendimento deste trabalho. A Figura 1 ilustra como os conceitos estudados se relacionam para compor o embasamento para a especificação da abordagem proposta, onde é possível identificar as relações que a abordagem dirigida por modelos busca alcançar, atuando no conceito de IoT e computação em nuvem.

Figura 1 – Relação entre os conceitos da fundamentação teórica



Fonte: Elaborado pelo autor.

Este capítulo está organizado em 4 seções. A Seção 2.1 apresenta a abordagem de desenvolvimento dirigida por modelos. Em seguida, a Seção 2.2 introduz os conceitos de IoT. A Seção 2.3 descreve os principais conceitos da computação em nuvem. Por fim, a Seção 2.4 apresenta o paradigma CoT.

2.1 Desenvolvimento Dirigido por Modelos

As disciplinas de engenharia têm a atividade de modelagem como uma técnica fundamental para lidar com a complexidade. A modelagem fornece uma maneira de facilitar a compreensão, o raciocínio, a construção, a simulação e a comunicação sobre sistemas complexos (THOMAS, 2004).

A aplicação de modelos para desenvolvimento de *software* tornou-se ainda mais popular com o desenvolvimento da Linguagem de Modelagem Unificada (do inglês, *Unified Modeling Language* - UML). No entanto, a relação entre a implementação do modelo e do *software* é apenas intencional sendo tratado como uma mera documentação. Dessa forma, apresenta duas desvantagens graves: por um lado, os sistemas de *software* não são estáticos e são susceptíveis a mudanças significativas, especialmente durante as primeiras fases do seu ciclo de vida e a documentação precisa ser adaptada, o que pode ser uma tarefa complexa e custosa. Por outro lado, esses modelos nem sempre representam o código implementado, uma vez que depende da interpretação do desenvolvedor de software. O desenvolvimento dirigido por modelos possui uma abordagem totalmente diferente, os modelos não constituem apenas a documentação, mas são considerados iguais ao código (VÖLTER et al., 2013).

A abordagem dirigida por modelos (do inglês, *Model Driven Development - MDD*) permite uma abstração dos detalhes da tecnologia da plataforma de implementação e utilização de conceitos mais próximos do domínio do problema. Consequentemente, os modelos utilizados para desenvolver o sistema são mais fáceis de especificar, entender e manter, e são resistentes às alterações da tecnologia adotada para a implementação do sistema. Assim, o MDD promove a ideia de que através do foco nos modelos pode-se obter melhor desenvolvimento e evolução do *software* (SELIC, 2003).

O MDD foca na representação do conhecimento através de modelos de domínios que por sua vez permitem a separação dos detalhes do problema e da implementação e por fim, esses modelos podem ser transformados em códigos executáveis (HERFS et al., 2013).

A proposta do MDD é de que o engenheiro de software se concentre em modelos de alto nível, ao contrário de interagir diretamente com o código-fonte, protegendo-se da complexidade das implementações de cada plataforma. Desta forma modelos passam a fazer parte do *software* e não meramente como artefatos de documentação. Segundo LUCRÉDIO (2009) as principais vantagens do uso de abordagens dirigidas por modelos são:

- **Produtividade:** melhoria no tempo de desenvolvimento. Tarefas repetitivas são implementadas nas transformações, poupando tempo e esforço que podem ser usados em tarefas mais importantes;
- **Portabilidade:** códigos podem ser gerados para diferentes plataformas a partir de um mesmo modelo;
- **Interoperabilidade:** cada parte do modelo pode ser transformado em código para uma

plataforma diferente e podem ser gerados adaptadores independentes de plataforma para que os códigos possam se comunicar;

- **Manutenção e Documentação:** em processos convencionais os modelos são utilizados como artefato inicial e depois são abandonados. Já em MDD os modelos fazem parte do *software*, então são mantidos atualizados, o que facilita as tarefas de manutenção;

- **Comunicação:** modelos podem ser utilizados para facilitar a comunicação entre os envolvidos no projeto. Os especialistas de domínio participam mais ativamente do processo, utilizando modelos para identificar os conceitos de negócio. Os especialistas de computação podem utilizar dos mesmos identificando os aspectos técnicos.

- **Reutilização:** Os modelos podem ser reutilizáveis em diferentes projetos, onde o próprio código-fonte pode ser re-gerado para o novo contexto;

- **Verificações e otimizações:** verificadores semânticos podem ser utilizados para reduzir a ocorrência de erros. Fornecendo implementações mais eficientes;

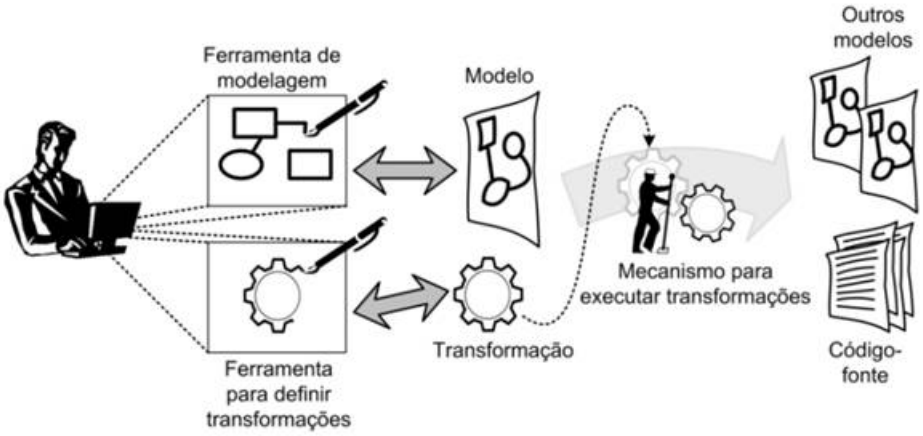
- **Corretude:** geradores de código podem evitar erros de digitação ou outros que poderiam ser introduzidos por um desenvolvedor.

Para possibilitar a criação de modelos, é necessária uma ferramenta de modelagem. Utilizando essa ferramenta, o engenheiro de *software* produz modelos que descrevem os conceitos do domínio. Para isto, a ferramenta deve ser intuitiva e de fácil utilização. Ao mesmo tempo, os modelos por ela criados precisam ser semanticamente completos e corretos, uma vez que devem poder ser compreendidos por um computador, e não um ser humano, capaz de corrigir pequenos enganos ou preencher lacunas por si só.

Os modelos servem de entrada para transformações que irão gerar outros modelos ou o código-fonte. Para definir as transformações, é necessária uma ferramenta que permita que o engenheiro de software construa regras de mapeamento de modelo para modelo, ou de modelo para código. Idealmente, essa ferramenta deve possibilitar que as regras de mapeamento sejam construídas da forma mais natural possível, uma vez que a construção de transformadores é uma tarefa complexa.

Finalmente, é necessário um mecanismo que efetivamente aplique as transformações definidas pelo engenheiro de *software*. Esse mecanismo deve não só executar as transformações, mas também manter informações de rastreabilidade, possibilitando saber a origem de cada elemento gerado, seja no modelo ou no código-fonte. A Figura 2 mostra os principais elementos necessários para uma abordagem MDD, e como eles interagem entre si.

Figura 2 – Principais elementos do MDD



Fonte: LUCRÉDIO (2009)

O Grupo de Gerenciamento de Objetos (do inglês, *Object Management Group* - OMG) é um dos responsáveis pelo atual aumento de interesse nas ideias do MDD, devido principalmente à Arquitetura Dirigida a Modelos (do inglês, *Model Driven Architecture* - MDA). A MDA surgiu como um conjunto de padrões voltados para fabricantes de ferramentas interessados em manter a interoperabilidade com outros fabricantes.

A MDA oferece uma abordagem aberta, neutra de fornecedor, para o desafio da mudança de negócios e tecnologia. A especificação MDA coloca ênfase em um processo em camadas, usando diferentes pontos de vista, que fornecem maneiras de representar funcionalidades do sistema através de interfaces e padrões de design específicos, que caracterizam o comportamento e os processos de negócios de qualquer aplicativo sem se preocupar com detalhes técnicos da uma plataforma específica.

O MDA propõe três tipos de pontos de vista: (i) Modelo Independente de Computação (do inglês, *Computation Independent Model* - CIM), (ii) Modelo Independente da Plataforma (do inglês, *Platform Independent Model* -PIM) e (iii) Modelo Específico da Plataforma (do inglÊs, *Platform Specific Model* - PSM). Esses modelos devem ser legíveis por máquinas para que eles sejam sucessivamente transformados em esboços de códigos, esquemas, padrões de teste e *scripts* de implantação para diversas plataformas (KLEPPE et al., 2003).

Um CIM é uma visão do sistema de um ponto de vista que não depende de computação. Um CIM não mostra detalhes da estrutura dos sistemas. É também conhecido como modelo do domínio, e utiliza um vocabulário familiar aos profissionais e especialistas no domínio em questão.

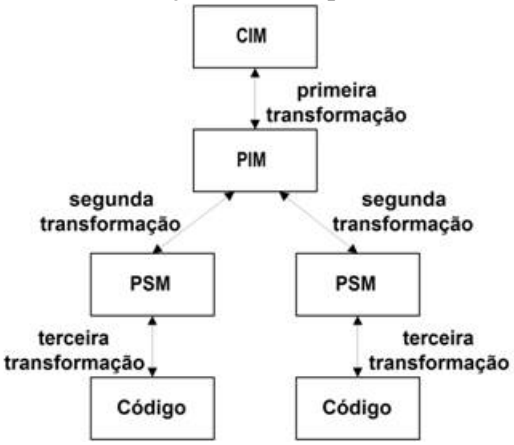
Um PIM é uma visão do sistema de forma independente da plataforma de implementação. Essa independência de plataforma, no entanto, chega até um certo nível, de forma que o mesmo modelo possa ser reaproveitado em diferentes plataformas do mesmo tipo.

Um PSM é uma visão do sistema do ponto de vista de uma plataforma específica. Um PSM combina as especificações de um PIM com detalhes sobre como o sistema utiliza aquele tipo

particular de plataforma.

A Figura 3 apresenta as transformações propostas pelo MDA, essas transformações são utilizadas para transformar um modelo em outro, sucessivamente, até o código-fonte. A transformação entre CIM e PIM é menos passível de automação, pois envolve mais decisões e maiores possibilidades de interpretação dos requisitos. Já a transformação entre PSM e código-fonte é mais passível de automação, já que o PSM está intimamente ligado com a plataforma de implementação.

Figura 3 – Transformações entre os pontos de vista do MDA



Fonte: Elaborado pelo autor.

A MDA também define o XMI (Intercâmbio de Metadados em XML, ou do inglês, XML Metadata Interchange), um formato para representar modelos em XML. Este formato define uma estrutura de documento que considera a relação entre os dados e seus correspondentes metadados. Assim, é possível para uma ferramenta, ao interpretar este formato, identificar quais os metadados que descrevem os dados sendo lidos. Por ser baseado em XML, traz consigo várias vantagens, tais como a facilidade de ser interpretado e a possibilidade de se aplicar transformações.

Apesar de ter a proposta de ser genérica, podendo trabalhar com qualquer linguagem de modelagem, a MDA tem grande foco na UML. Por exemplo, perfis UML são o mecanismo preferido para representação de conceitos específicos de plataforma em um PSM.

2.1.1 UML

A UML (RUMBAUGH; JACOBSON; BOOCH, 2004) é uma linguagem visual que é aplicada para a especificação, construção e documentação de artefatos de software. Trata-se de uma linguagem de modelagem com propósito geral que pode ser usada com o paradigma orientado a objetos ou a componentes. Além disso, a linguagem caracteriza-se pela aplicabilidade em diferentes domínios de aplicações e plataformas de implementação, a UML tem se tornado uma linguagem de modelagem padrão, sendo usada tanto na indústria quanto na academia.

Pelo fato da UML ser uma linguagem de caráter geral e ter um escopo de aplicação muito grande, em determinadas situações não é capaz de modelar aplicações definidas em um domínio específico (por exemplo, área financeira, aplicações de tempo real, aplicações multimídia, entre outros). Isto caracteriza a situação onde a sintaxe ou a semântica especificada na UML não é capaz de expressar conceitos de um sistema em particular, ou quando deseja-se tornar os elementos já existentes mais restritos, ou seja, diminuir o seu escopo. Sendo assim, surge a necessidade de criar um extensão da linguagem.

A OMG estabelece duas formas de definições de linguagem específicas de domínios. A primeira é baseada na definição de uma nova linguagem,sendo esta uma alternativa a UML. Para isto, é utilizado os mecanismos fornecidos pela OMG para definição de linguagens visuais orientadas a objetos. Esta abordagem representa a mesma forma utilizada para a definição da UML. Sendo assim, a sintaxe e a semântica dos elementos na nova linguagem são definidas para atender as características específicas do domínio.

A segunda alternativa consiste na especialização da UML. Para isto, alguns elementos definidos da UML são estendidos através das definições de restrições sobre os mesmos. Porém, com esta abordagem, o metamodelo da UML é respeitado, deixando a semântica original dos elementos da UML inalterados (por exemplo, as propriedades de classes, operações e entre outros elementos permanecem inalteradas). Novas notações podem ser definidas a fim de uma melhor representação dos conceitos.

A extensão do metamodelo tem o objetivo de aumentar o poder de representação da UML e pode ser alcançado com o uso de perfis. Os perfis podem sofrer especializações com o objetivo de satisfazer as necessidade e as exigências de um subdomínio. Para isto, a UML fornece mecanismos de refinamento, sendo eles: (i) valores marcados; (ii) estereótipos; e (iii) restrições. Estes elementos podem ser usados para adaptar a semântica da UML, sem alterar o metamodelo da UML de fato. Uma vez definidos são agrupados em um perfil.

Existem várias razões para fazer o uso dos perfis, tais como: (i) fazer uso de terminologias que são adaptadas para um domínio específico ou uma plataforma; (ii) atribuir notações diferentes ou alternativas para elementos definidos no metamodelo de referência; (iii) adicionar semântica que não são especificadas; (iv) adicionar restrições limitando o uso do metamodelo e de seus construtores; (v) inserir informações que podem ser usadas quando o modelo é transformado em outro modelo ou em código.

O propósito dos perfis UML é permitir a construção e permutação de modelos UML que exijam especificação semânticas, além das quais podem ser representados com a UML padrão. Observando que as semânticas adicionadas são completamente consistentes com a semântica geral da UML. Por exemplo, novas restrições associadas aos estereótipos não podem contradizer as restrições do metamodelo da UML, mas podem torná-las mais restritas ou especializadas, realizando uma extensão conservativa.

2.2 *Internet* das Coisas

O conceito de *Internet* das Coisas (do inglês, *Internet of Things* - IoT), refere-se à conexão de vários objetos cotidianos ("coisas") à *Internet*, que coletam e transmitem informações para aplicações a fim de monitorar e atuar no ambiente (AL-FUQAHA et al., 2015).

IoT tem um grande impacto na vida de seus usuários, a disponibilidade de informações antes inexistentes trazem muitas oportunidades para um grande número de aplicações inovadoras que prometem melhorar a qualidade de nossas vidas (XIA et al., 2012). Áreas como automação residencial, assistência na saúde e aprendizado virtual são exemplos de áreas domésticas onde a IoT terá grande impacto. No campo de atividades comerciais, podemos listar automação industrial, logística, transporte inteligente, dentre outros (ATZORI; IERA; MORABITO, 2010).

IoT aumentará a onipresença da *Internet* integrando cada objeto para interação através de sistemas incorporados, o que leva a uma rede de dispositivos altamente distribuída que comunicam-se com seres humanos e com com outros dispositivos. É previsto que hajam bilhões de objetos conectados, fornecendo uma quantidade enorme de informação em diversas áreas de aplicação (GIUSTO et al., 2010).

A evolução da IoT se deve ao avanço das tecnologias de dispositivos, protocolos, redes e a virtualização de computadores. Assim sendo vários elementos interligados formam um sistema que visa um consumo de dados em tempo real. Segundo (AL-FUQAHA et al., 2015) os principais elementos que compõem uma aplicação de IoT são:

- **Identificação:** é crucial para identificar os objetos de forma única para conectá-los à internet. Muitos métodos são utilizados para identificar, como: RFID (*Radio Frequency Identification*), NFC (*Near Field Communication*), endereço IP. Esses métodos de identificação são usados para fornecer uma identidade clara para cada objeto dentro da rede.

- **Detecção:** possuem sensores e atuadores que coletam informações do ambiente e podem realizar ações específicas com base nos serviços requeridos.

- **Comunicação:** tecnologias de comunicação da IoT conectam objetos heterogêneos para fornecer serviços inteligentes específicos. Os diversos dispositivos devem operar com pouca energia e falhas de rede podem ser comuns. A partir desses problemas, a escolha da tecnologia de comunicação deve ser considerada relevante ao projeto.

- **Computação:** unidades de processamento e aplicativos de *software* que representam a capacidade e a inteligência da IoT. As plataformas de nuvem formam outra parte computacional importante da IoT. Essas plataformas fornecem recursos para que os objetos inteligentes enviem seus dados para a nuvem, para que os dados sejam processados em tempo real e, para que os usuários finais se beneficiem do conhecimento extraído dos dados coletados.

- **Serviços:** os serviços de IoT podem ser categorizados em quatro classes: Serviços de identidade pode ser considerados um dos mais importantes e é utilizado em outros tipos de serviços. Toda aplicação precisa trazer objetos do mundo real para o virtual e é necessário identificar essas entidades. Agregação de informação consiste em serviço que coleta e ordena os dados recebidos pelos dispositivos para que sejam processados e reportado na aplicação IoT. Os Serviços de Colaboração e Inteligência consiste em agir sobre os serviços de agregação de informações usando os dados obtidos para tomar decisões para assim reagir de modo adequado. Os Serviços de Ubiquidade tem como objetivo prover Serviços de colaboração e Inteligência em qualquer momento e qualquer lugar em que eles sejam necessários. O objetivo final de todos os aplicativos da IoT é alcançar o nível de serviços onipresentes. No entanto, este fim não é possível facilmente, uma vez que existem muitas dificuldades e desafios que devem ser abordados.

- **Semântica:** habilidade de extrair conhecimento de diferentes provedores de serviços. Tal conhecimento, incluem descobrimento, uso de recursos e modelagem de informação. Também inclui o reconhecimento, análise do dado para fazer com que as decisões sejam corretas.

Recentemente, uma nova era de IoT chamada IoT Cognitiva foi enunciada. Ele visa integrar tecnologias cognitivas em sistemas baseados em IoT para garantir o gerenciamento inteligente, permitindo a cooperação e interação entre IoT e humanos. A computação autônoma de acoplamento com computação cognitiva abrange oportunidades sem precedentes para o desenvolvimento de sistemas inteligentes baseados em IoT (FOTEINOS et al., 2013).

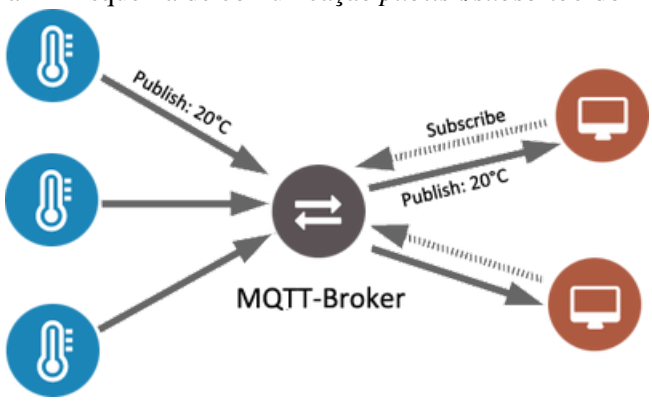
O Consórcio da Internet Industrial (do inglês, Industrial Internet Consortium) afirma que a IoT, construída sobre a nuvem, *internet* móvel, dados importantes, etc., é chamada de revolução tecnológica, sendo a terceira transformação fundamental após a Revolução Industrial e a Revolução da *Internet*. Esse consórcio foi fundado em março de 2014 (pela ATT, Cisco, General Electric, IBM e Intel) para remover obstáculos à adoção generalizada do IoT. Sua missão é acelerar o crescimento da *Internet* Industrial, coordenando iniciativas de ecossistemas para conectar e integrar objetos com pessoas, processos e dados usando arquiteturas comuns, interoperabilidade e padrões abertos que levam a resultados de negócios transformacionais (ANNUNZIATA; EVANS, 2012).

Para a implementação de IoT surgiram alguns protocolos específicos como MQTT (*Message Queuing Telemetry Transport*) e CoAP (*Constrained Application Protocol*). O MQTT é um protocolo de transferência de mensagens leve e simples, ele utiliza o paradigma *publisher/subscriber* para troca de mensagens e implementa um *middleware* chamado de MQTT *broker* que é responsável por receber, enfileirar e distribuir as mensagens recebidas dos *publishers* para os *subscribers* (MESNIL, 2014).

O paradigma pub/sub utiliza o conceito de tópicos para processar as mensagens, onde cada mensagem é enviada para um determinado tópico. Diferente de outros protocolos de mensagem,

o *publisher* não envia a mensagem diretamente ao *subscriber*, mas sim ao *broker*. O *publisher* envia a mensagem para o MQTT *broker* em um determinado tópico. O *broker* é responsável por receber a mensagem do *publisher* e fazer uma pré-filtragem das mensagens e enviá-las para os subscribers que estiverem registrados em um determinado tópico (LAMPKIN et al., 2012).

Figura 4 – Esquema de comunicação *publish/subscribe* do MQTT.



Fonte: Elaborado pelo autor.

2.3 Computação em Nuvem

A Computação em Nuvem é um modelo que permite o acesso a uma rede compartilhada, conveniente e *on-demand*, a um *pool* compartilhado de recursos de computação configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente provisionados e lançados com o mínimo esforço de gerenciamento ou interação do provedor de serviços.

Inicialmente o ambiente de computação em nuvem era composto de três modelos de serviços: *Software* como um Serviço (do inglês, *Software as a Service* - SaaS), Plataforma como um Serviço (do inglês, *Platform as a Service* - PaaS) e Infraestrutura como um Serviço (do inglês, *infraestructure as a Service* - IaaS). A partir deles se definiam o padrão arquitetural para soluções de computação em nuvem (ARMBRUST et al., 2009).

- **Infraestrutura como um Serviço:** O modelo IaaS oferece uma infraestrutura de processamento e armazenamento de forma transparente, onde o usuário não tem o controle da infraestrutura física, mas através de mecanismos de virtualização, possui controle sobre as máquinas virtuais, armazenamento, aplicativos instalados e um controle limitado dos recursos de rede (VERAS, 2012).

- ***Software* como um Serviço:** No modelo SaaS, aplicativos de interesse para uma grande quantidade de clientes passam a ser hospedados na nuvem como alternativa ao processamento local. Os aplicativos são oferecidos como serviços por provedores e acessados pelos clientes por aplicações como navegadores *Web*. Todo o controle e gerenciamento

da rede, sistemas operacionais, servidores e armazenamento é feito pelo provedor de serviços (VERAS, 2012).

- **Plataforma como um Serviço:** No modelo PaaS são oferecidos recursos e ferramentas pelo provedor voltados para facilitar a produtividade do desenvolvedor de aplicativos, que serão executados e disponibilizados diretamente na nuvem. A plataforma na nuvem oferece um modelo de computação, armazenamento e comunicação para os aplicativos (VERAS, 2012).

Os conceitos de computação em nuvem ainda se aprimoram, mas a proposta principal é criar a ilusão de recursos computacionais infinitos e eliminar o comprometimento antecipado de capacidade, além disso permitir o pagamento pelo uso real dos recursos (VERAS, 2012).

Recentemente surgiu um novo modelo para a implantação de aplicativos em nuvem, em grande parte devido à recente mudança de arquiteturas de aplicativos empresariais para contêineres e micro serviços, chamado de Computação sem Servidor.

Esse novo modelo proporciona a utilização de algum serviço sem a necessidade de um sistema operacional contido em um servidor físico. Consiste em uma combinação de serviços sendo utilizados como *hosts* remotos que são executados como funções, abstraindo totalmente a questão de qualquer configuração, manutenção e administração pelo desenvolvedor, ou seja, o próprio fornecedor dos serviços tem essa responsabilidade.

A engenharia *Cloudware* emprega abordagens sistemáticas, disciplinadas e quantificáveis para o desenvolvimento, operação e manutenção de um tipo específico de *software*: aplicações nativas da nuvem. A entrega de tais aplicações exigem que revisemos teorias, métodos, arquiteturas, tecnologias e ferramentas de engenharia de *software* tradicionais para garantir a entrega de aplicativos escaláveis, elásticos, portáteis, resilientes e adaptáveis para a nuvem.

2.4 Nuvem das Coisas

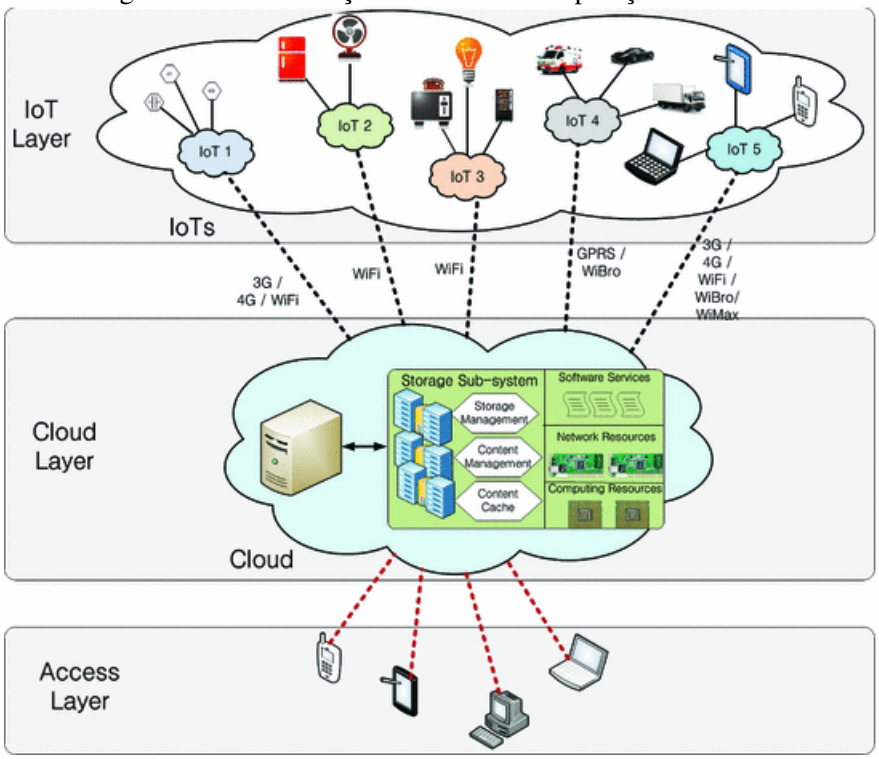
Aplicações de IoT abrangem todos os níveis de complexidade de *software* muito simples que funcionam em sensores básicos e outros dispositivos simples para os sistemas escaláveis, de alto desempenho, confiáveis, altamente governados, resistentes, resilientes, necessários para processar, analisar e responder às enormes quantidades de dados que produzem (ZHOU et al., 2013).

Está se tornando muito difícil gerenciar sensores pequenos com restrição de energia e outros dispositivos geradores de dados. Com as IoTs, qualquer coisa pode se tornar parte da Internet e gerar dados. Além disso, os dados gerados precisam ser gerenciados de acordo com seus requisitos, a fim de criar serviços mais valiosos (AAZAM et al., 2016).

Assim, existe a necessidade de um ambiente forte e flexível para o suporte das aplicações de IoT. A computação em nuvem fornece uma base sólida para o compartilhamento de recursos de forma flexível. Para esse propósito, a integração de IoT com a computação em nuvem está se

tornando muito importante. Este novo paradigma é denominado Nuvem das Coisas (do inglês, *Cloud of Things* - CoT). CoT fornece meios para lidar com dados crescentes e outros recursos de IoT (AAZAM et al., 2016). A Figura 5 apresenta o esquema de comunicação de CoT.

Figura 5 – Comunicação entre IoT e computação em nuvem



Fonte: (AAZAM et al., 2016)

3 TRABALHOS RELACIONADOS

Neste capítulo será realizada uma análise comparativa de trabalhos que possuem relação com o objetivo e questões de pesquisa (Seção 1.2) deste trabalho. Esses trabalhos são publicações de revistas, periódicos e conferências especializadas da área de computação e foram selecionados seguindo alguns critérios que serão apresentados nas próximas seções.

Ao final deste capítulo será possível: (a) obter uma visão geral em relação ao estado da arte sobre as abordagens dirigidas por modelos no desenvolvimento de CoT, (b) detalhar as contribuições já identificadas e pontos de melhorias nesse tipo de abordagem, (c) compreender melhor o foco da pesquisa deste trabalho, bem como (d) comparar as contribuições deste trabalho com os encontrados na literatura.

Este capítulo está organizado da seguinte maneira: A Seção 3.1 descreve a metodologia para seleção dos trabalhos relacionados. A Seção 3.2 apresenta os trabalhos selecionados e descreve suas técnicas e ferramentas. A Seção 3.3 apresenta uma comparação entre os trabalhos relacionados com base nos critérios comparativos definidos. Por fim, a Seção 3.4 apresenta as oportunidades de pesquisas identificadas após a análise da literatura.

3.1 Metodologia para a Escolha dos Trabalhos

Algumas abordagens dirigidas por modelos no desenvolvimento de aplicações, incluindo as aplicações de CoT, já vem sendo discutidas na literatura. Este trabalho busca realizar uma investigação desses trabalhos, a fim de identificar lacunas relacionadas ao desenvolvimento de aplicação CoT através de uma abordagem baseada em modelos.

Para o processo de seleção dos trabalhos foram definidas quatro etapas principais, sendo elas: (a) pesquisa em bases científicas através de termos de buscas relacionados ao estudo primário do trabalho proposto; (b) leitura do título, palavras-chave e resumo; (c) leitura da introdução e conclusão; e, por fim (d) leitura completa dos artigos.

Para fins de seleção dos trabalhos foram considerados publicações na área da computação, selecionados nas principais bibliotecas digitais conforme apresentado na Tabela 1, que se destacam por serem amplamente utilizadas para pesquisas na área da computação, sendo estas reconhecidas por sua abrangência e relevância significativa na área de estudo.

Tabela 1 – Bibliotecas digitais consultadas por esta pesquisa

Biblioteca Digital	**Endereço *Web***
ACM Digital Library	https://dl.acm.org/
Google Scholar	https://scholar.google.com.br/
IEEE Xplore Digital Library	https://ieeexplore.ieee.org
Science Direct	https://www.sciencedirect.com/
Springer Link	https://link.springer.com/

Fonte: Elaborado pelo autor.

Para realizar a busca inicial nas bibliotecas citadas, reunindo os principais termos relacionados com a pesquisa (Desenvolvimento Dirigido por Modelos, *Internet* das Coisas e Computação em Nuvem), chegou-se a elaboração da seguinte *string* de busca:

(“MDD"**OR** “MDA"**OR** “MODEL DRIVEN DEVELOPMENT"**OR** “MODEL DRIVEN ARCHITECTURE") **AND** (“INTERNET OF THINGS"**OR** “IOT"**OR** “CLOUD OF THINGS"**OR** “CLOUD COMPUTING”).

A partir dos critérios de busca e dos filtros aplicados, chegou-se ao total de 8 trabalhos que foram selecionados para compor um estudo mais detalhado, a fim de realizar uma análise comparativa entre eles. Para isso, a Seção 3.2 descreverá resumidamente cada trabalho selecionado. A Seção 3.3 apresentará os critérios utilizados para comparação dos trabalhos a fim de identificar as oportunidades de pesquisa que serão apresentadas na Seção 3.4.

3.2 Análise dos Trabalhos Selecionados

Nesta seção, serão descritos resumidamente os 8 trabalhos selecionados para o estudo detalhado, os quais possuem como foco abordagens dirigidas por modelos para o desenvolvimento de aplicações para IoT e/ou serviços de computação em nuvem.

3.2.1 A model-driven methodology for the design of autonomic and cognitive IoT-based systems: application to healthcare

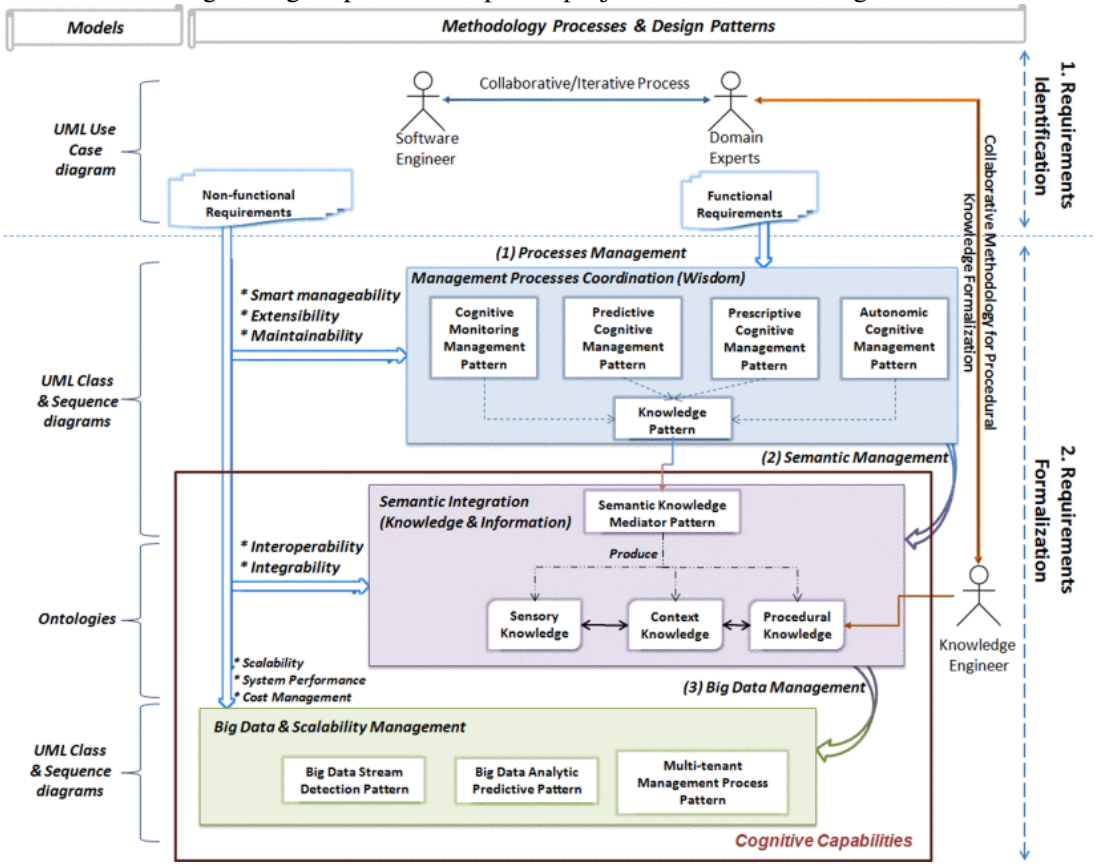
MEZGHANI; EXPOSITO; DRIRA (2017) propuseram uma metodologia dirigida por modelo que combina princípios de modelagem de *software* e engenharia de conhecimento para facilitar o projeto e o desenvolvimento de sistemas baseados em IoT autônomos e cognitivos. Conforme apresentado na Figura 6 são identificadas duas fases principais: Identificação de Requisitos e Formalização de Requisitos.

A identificação dos requisitos é baseada em discussões com os especialistas do domínio, a fim de identificar os requisitos funcionais e não funcionais do sistema. É um processo iterativo, onde os requisitos são aperfeiçoados e representados usando diagramas de casos de uso UML, que descrevem as funções do sistema sem especificar detalhes de implementação. A próxima fase, que é a formalização dos requisitos, enfoca a formalização e estruturação dos requisitos identificados em modelos concretos que descrevem as interações dos processos do sistema.

Dentro da fase de formalização de requisitos, apresentam três subníveis para lidar de forma incremental com os desafios relacionados ao projeto: o primeiro nível é a coordenação dos processos de gestão, nesse nível identificaram cinco padrões que consideram a capacidade de gerenciamento inteligente, extensibilidade e manutenção de sistemas baseados em IoT. Esses padrões são representados com base em diagramas de classes para projetar a estrutura e diagramas de sequência para desenhar o comportamento e delinear como os processos de gerenciamento devem ser coordenados e interagir para atender aos requisitos funcionais do sistema.

Uma vez que os processos de gerenciamento são identificados e modelados, o próximo nível é a integração semântica, onde principalmente as informações sobre o sistema e seu ambiente, bem como o conhecimento para a tomada de decisões, são formalizadas para serem reutilizadas automaticamente pelos processos de gestão. Dentro desse nível, os autores identificaram um padrão para garantir a interoperabilidade e a integração para o bom funcionamento do sistema. A aplicação deste padrão leva à produção de três tipos de modelos semânticos (o sensorial, o contexto e o conhecimento processual) que são projetados com base em ontologias. Finalmente, o último nível onde definiram três padrões que lidam com os grandes desafios de dados (volume, variedade, velocidade), escalabilidade, desempenho do sistema e gerenciamento de custos do sistema.

Figura 6 – Metodologia dirigida por modelo para o projeto de sistemas inteligentes baseados em IoT



Fonte: MEZGHANI; EXPOSITO; DRIRA (2017)

Os autores utilizaram a metodologia proposta para projetar e desenvolver um sistema de monitoramento inteligente capaz de gerenciar a evolução da saúde dos pacientes com base em dispositivos portáteis que medem o açúcar no sangue, os requisitos funcionais deste sistema englobaram a visualização e detecção das anomalias de saúde do paciente e interação com o médico apropriado. O objetivo da avaliação foi destacar a capacidade do sistema para gerenciar a variedade de dados, volume e velocidade, bem como sua escalabilidade e desempenho com base nos padrões propostos. Citam como trabalhos futuros, elaborar um conjunto de regras de

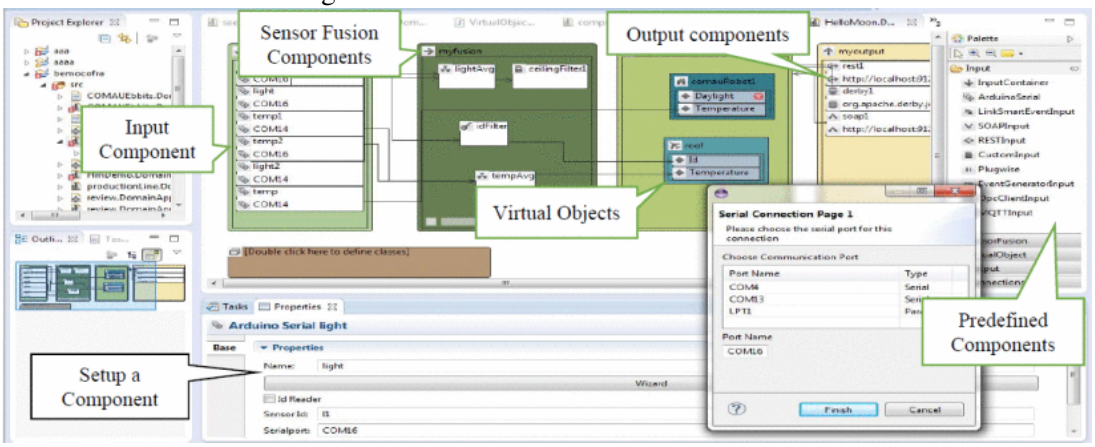
transformação que refinem automaticamente o *design* do sistema baseado em IoT com base em uma especificação dos requisitos do sistema e também trabalhar no fornecimento de melhores configurações de recursos de TI para otimizar o tempo de resposta.

O trabalho dos autores MEZGHANI; EXPOSITO; DRIRA (2017), por mais que apresentam uma metodologia para processar grandes volumes de informação em um ambiente escalonável, não apresentam outros recursos de aplicações CoT e não implementaram ferramentas para automatizar o processo de criação da aplicação.

3.2.2 Connecting the Internet of Things rapidly through a model driven approach

PRAMUDIANTO et al. (2016) propuseram uma ferramenta de desenvolvimento para IoT, chamada IoTLink que foi projetada para oferecer suporte a desenvolvedores inexperientes conectando componentes IoT rapidamente através de uma abordagem de desenvolvimento dirigida por modelo que eleva o nível de abstração, o que reduz a complexidade dos artefatos de *software* e os esforços necessários para produzi-los. Uma vez definido o modelo, a ferramenta é capaz de gerar os artefatos do *software*, representando os objetos físicos. A Figura 7 apresenta a interface do IoTLink.

Figura 7 – Interface do usuário do IoTLink



Fonte: PRAMUDIANTO et al. (2016)

Sua arquitetura consiste em cinco camadas, a camada de aplicação abstrai as aplicações ou a lógica de negócio que acessam serviços de IoT através da camada de saída. A camada do modelo de domínio é responsável por virtualizar os dispositivos físicos que são relevantes para as aplicações. Na camada mais inferior, os dispositivos são gerenciados e abstraídos com uma interface comum que é conhecida pelas camadas superiores, e pôr fim a camada de fusão fornece módulos genéricos de pré-processamento para extrair informações sobre os objetos das camadas mais baixas.

Com base na arquitetura proposta, o IoTLink permite que os usuários modelem a representação de objetos físicos, incluindo seus atributos e funções. Os atributos podem ser vinculados

a sensores e os componentes de entrada podem ser vinculados aos módulos de fusão do sensor quando é necessário processamento de dados para determinar o estado dos objetos de domínio. Com base no modelo ele pode gerar o código Java e quando esse código é compilado e executado, ele pode atuar como *proxy* para os objetos físicos envolvidos no domínio do aplicativo.

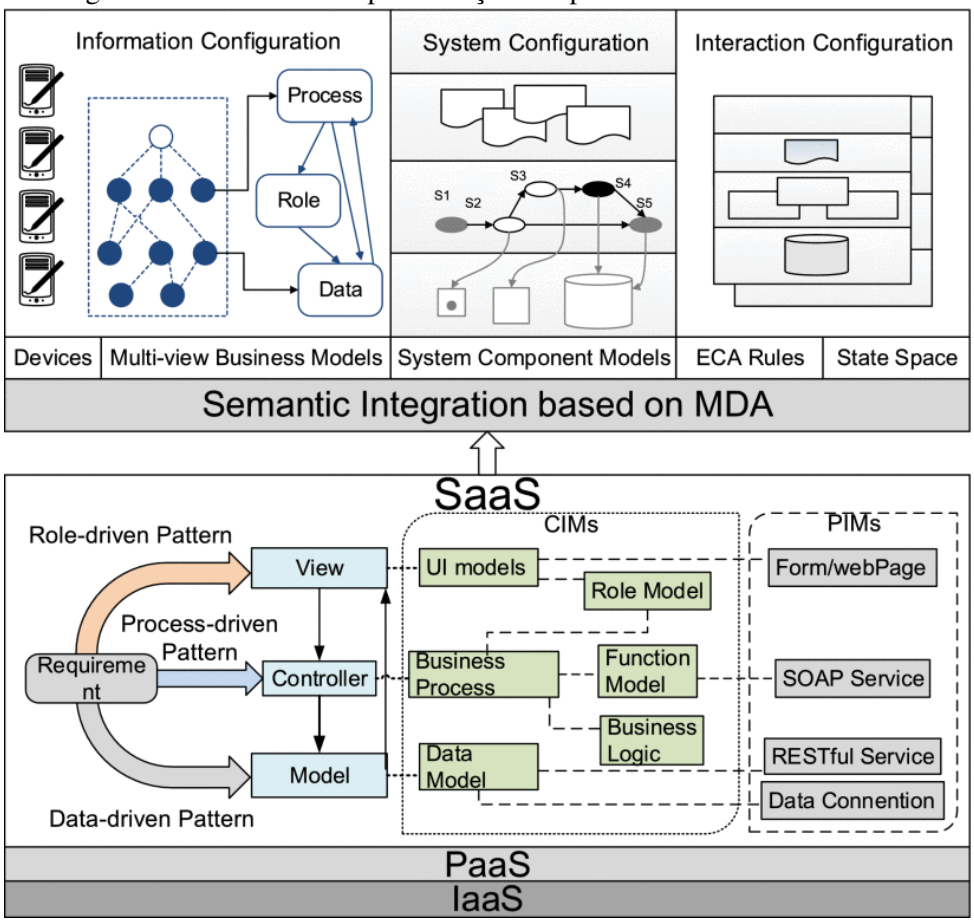
Um estudo foi projetado para identificar os efeitos na compreensão do programa e no tempo necessário para implementar e modificar o *software* pretendido. Como linha de base, foi comparado com a eficiência e eficácia de abordagens amplamente utilizadas, incluindo UML e Java. Os autores concluíram que os diagramas são capazes de apresentar relações gerais de forma mais eficiente do que o código-fonte. No entanto, à medida que os diagramas se tornam mais complexos, eles exigem mais tempo para analisar do que um texto. Muitos participantes envolvidos se mostraram positivos quanto a nova abordagem para o desenvolvimento de IoT, que mostra ter um enorme potencial para permitir a prototipagem rápida, pois algumas etapas podem ser automatizadas por geração de código, reduzindo significativamente o tempo de desenvolvimento e mantendo a qualidade do código.

O trabalho de PRAMUDIANTO et al. (2016) apresenta uma abordagem à nível de dispositivo de IoT, e não envolve funcionalidades de computação em nuvem.

3.2.3 Model-driven development patterns for mobile services in Cloud of Things

CAI et al. (2018) apresentaram uma estrutura de implementação de aplicativos móveis usando padrões de desenvolvimento por modelos. A estrutura pode ser dividida em três fases, conforme é mostrado na Figura 8:

Figura 8 – Estrutura de implementação de aplicativo móvel usando MDD



Fonte: CAI et al. (2018)

- **Configuração da informação:** com base no modelo de informação, a informação dos objetos de IoT é alcançada e representada de acordo com uma estrutura abstrata. Por meio de modelos multi-visões, uma visão unida relacionada a um determinado domínio é usada para gerenciar diferentes modelos.

- **Configuração do sistema:** de acordo com a arquitetura típica, os modelos de negócios são transformados em componentes de serviço, de modo a construir um sistema completo. Diferentes padrões são utilizados para realizar a configuração da informação e integração de processos em nível semântico.

- **Configuração de interação:** de acordo com os requisitos de negócios, a relação do cenário do aplicativo com as informações contextuais, que formam o ambiente de tempo de execução lógico é construído com base em recursos. Baseado no espaço de estados, as regras do ECA (Evento, Condição, Ação) são usadas para implementar a interação inteligente no ambiente de tempo de execução.

Durante o estágio de configuração de informações, o meta-modelo de múltiplas visualizações de negócios baseado em ontologia é fornecido para o gerenciamento de recursos. As regras

da ECA são usadas para definir ações com base nos recursos relacionados no estágio de configuração de interação. O espaço de estados de recursos relacionados é aplicado para construir um mecanismo de controle de execução no final do estágio de configuração de interação.

No estágio de configuração do sistema os CIMs relacionados são agrupados como três tipos de componentes, incluindo (a) componentes do tipo *User Interface*, como perfis do usuário,*Widgets* e *Gadgets*, (b) componentes do tipo controlador que representam o fluxo de controle da lógica de negócios, como o processo de negócios que consiste em funções, regra conectores, funções e relacionamento relacionado entre esses elementos, e (c) componentes de tipo de dados, como objetos de mapeamento de O/R e entidades de dados que são armazenados no banco de dados ou fontes de dados. E os PIMs em sistemas baseados na *Web* são representados como formulários/páginas da *Web*, conexões de dados, serviços SOAP e RESTful para serem conectados a componentes relacionados nos CIMs.

Com base nos recursos existentes de uma determinada plataforma de nuvem, a principal tarefa da MDA é concentrar-se na transformação de modelos de CIMs para PIMs e PSMs, de modo a desenvolver um sistema executável. Depois que os CIMs e PIMs são agrupados com base na estrutura MVC (*Model-View-Controller*) de referência, três padrões são propostos como elementos orientados relacionados aos requisitos de negócios de diferentes níveis.

- **Padrão orientado por função:** apresentado quando as funções e as interfaces de usuário são definidas primeiro como requisitos. Nessa situação, o Diagrama de Casos de Uso, que representa as relações funcionais, é usado no processo de desenvolvimento.

- **Padrão orientado pelo processo:** apresentado quando o processo de negócios é definido primeiro como requisitos. Nessa situação, os Diagramas de Processo, como BPMN (*Business Process Modeling Notation*) e EPC (*Event-Driven Process Chains*) e o Diagrama de Atividades, que representam fluxos de controle de tarefas, são usados no processo de desenvolvimento.

- **Padrão orientado por dados:** apresentado quando o modelo de dados é definido primeiro como requisitos. Em tal situação, Diagrama de Classes ou Diagrama de Entidades-Relação, que representa entidades e relações de informação, é usado no processo de desenvolvimento.

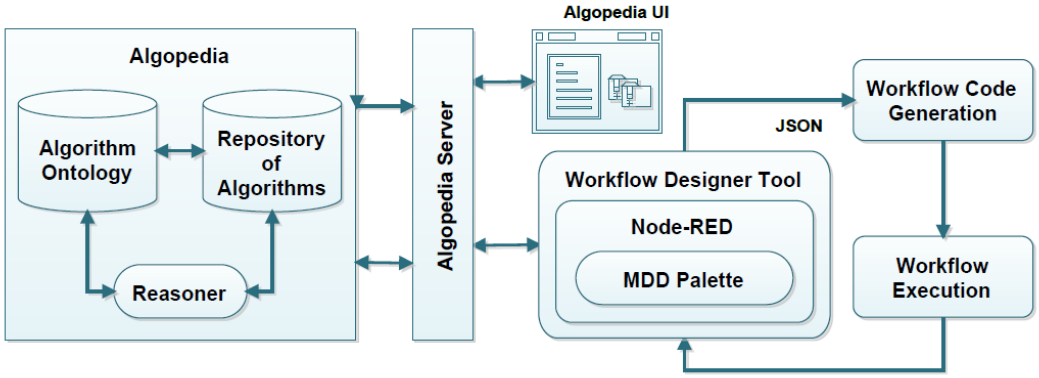
Com base nesses padrões uma plataforma é fornecida para o desenvolvimento para aplicações móveis na nuvem. Um estudo de caso de um sistema de gerenciamento de vendas é fornecido para ilustrar o processo de desenvolvimento de aplicativos móveis. Essa plataforma se mostrou mais adequada para implementar serviços em nuvem comparado a outras abordagens, por reduzir o custo de desenvolvimento para implementar serviços que envolvem relações complexas cobrindo processos, dados e funções com um alto nível de automação, por meio de configurações do cenário da aplicação de modo a cumprir os requisitos.

Os autores CAI et al. (2018) apresentaram um padrão de desenvolvimento indicado para serviços moveis que executam em computação em nuvem. No entanto não apresentam funcionalidades para integração de serviços de computação em nuvem com dispositivos de IoT.

3.2.4 A smart framework for IoT analytic workflow development

JAISWAL et al. (2015) propuseram um *framework* baseado em um estudo de caso que permite aos desenvolvedores arrastar e soltar algoritmos para criação de uma cadeia de fluxo de trabalho, selecionando automaticamente os recursos de sinais mais relevantes para a aplicação, usando um conjunto de dados para gerar um modelo e implantar esse modelo para uso. Uma visão geral do *framework* é apresentada na Figura 9.

Figura 9 – Fluxo de trabalho de alto nível para processamento de sinal de IoT



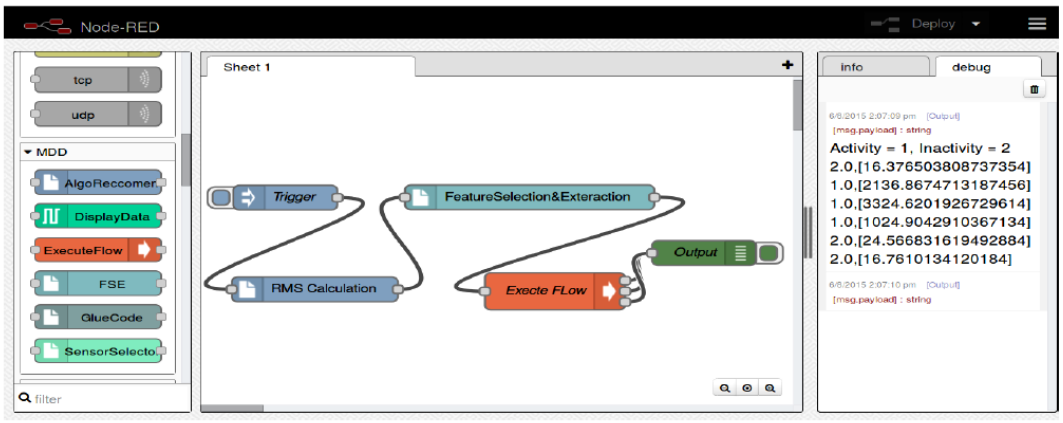
Fonte: JAISWAL et al. (2015)

Esse *framework* é construído em torno de um repositório de algoritmos que contém informações sintáticas e semânticas detalhadas de uma grande variedade de algoritmos usados no processamento de sinal do sensor para aplicações IoT. O *framework* permite que um desenvolvedor crie visualmente um fluxo de trabalho ao combinar as implementações de algoritmos e com base no tipo de entrada e no requisito de processamento específico, um modelo executável é gerado.

O sistema proposto tem duas fases, a primeira de criação do fluxo de trabalho para criar um modelo e a segunda para execução do fluxo de trabalho. O sistema fornece um conjunto de todos os recursos normalmente utilizados no processamento do sinal do sensor a partir do qual o usuário pode selecionar recursos ou optar pela seleção automática de características. Conforme a Figura 10 a implementação do sistema se baseia na plataforma Node-RED, que fornece uma maneira fácil de definir elementos personalizados, onde apresentam um painel separado contendo elementos correspondentes às etapas do fluxo de trabalho de processamento de sinal, como recomendação de algoritmo, seleção de características e extração e execução de fluxo de trabalho. Esses elementos se comunicam com o repositório de algoritmos para

recuperar a lista de algoritmos disponíveis, recursos que são disponibilizados ao usuário para seleção.

Figura 10 – Criação de um fluxo de trabalho no NODE-RED



Fonte: JAISWAL et al. (2015)

Esse ainda é um trabalho em andamento, mas considerando todos os pontos, essa estrutura poderá fornecer uma maneira econômica e eficiente para criar aplicativos analíticos IoT com base no processamento do sinal do sensor.

3.2.5 FRASAD: A framework for model-driven IoT application development

NGUYEN et al. (2015) propuseram um *framework* chamado FRASAD, que implementa uma abordagem dirigida por modelos e visa melhorar a reutilização, flexibilidade e manutenção do *software* de sensores. No nível de abstração mais alto da arquitetura proposta, um modelo baseado em regras e uma Linguagem Específica de Domínio (DSL) são usados para descrever os aplicativos. Foi elaborado para desacoplar a linguagem de programação e seu modelo de execução a partir do sistema operacional e *hardware* subjacente. Uma interface de usuário gráfica, componentes de geração de código e ferramentas de suporte foram incluídas para ajudar os desenvolvedores a projetar, implementar, otimizar e testar a aplicação IoT.

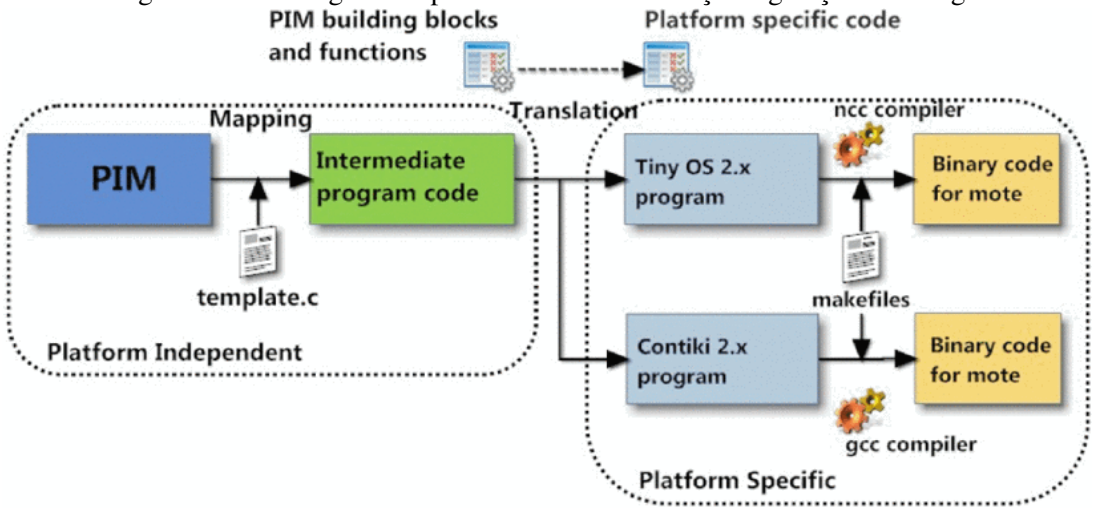
A arquitetura é projetada em uma abordagem de várias camadas, integrando a arquitetura tradicional dos nós de sensores e duas camadas superiores novas: a Camada de Aplicação (APL) e a Camada de Abstração do Sistema Operacional (OAL), que tem como objetivo oferecer um nível mais alto de abstração e portabilidade ao ocultar a plataforma inferior.

Para realizar a arquitetura proposta empregaram a abordagem MDA com várias camadas, onde cada camada pode interagir com a parte superior e inferior imediata através de uma interface pré-definida. Toda camada é encapsulada em um componente. O modelo independente de computação (CIM) é um modelo de domínio ou negócio que representa informações específicas do domínio, sem os detalhes da estrutura do sistema ou das tecnologias de implementação.

Na APL, um modelo independente de plataforma (PIM) é usado para capturar a lógica e os requisitos do aplicativo. Uma DSL foi projetada neste nível de abstração para suportar o processo de mapeamento. No OSL, é apresentado um conjunto de modelos de plataforma específicos (PSM) do sistema operacional correspondentes aos diferentes sistemas operacionais. Em seguida, dado um modelo de plataforma específica do sistema operacional, o OAL realiza a transformação do PIM para o PSM; portanto, esta camada pode ser vista como um gerador de aplicativos, que produz o código personalizado para implantar na plataforma de destino. Este PSM é usado pelo compilador para gerar o código binário a partir do código de programa recebido da camada superior.

A Figura 11 descreve o processo de transformação que leva o arquivo PIM (application.xml) como entrada e gera automaticamente o código equivalente para a plataforma de destino selecionada (sistema operacional e *hardware*). Primeiro, o arquivo PIM é mapeado para o código intermediário do programa (application.h, application.c) usando blocos e funções de criação de linguagem C (template.c) em combinação com um analisador XML. Na fase de tradução, o processo leva os arquivos de código de programa intermediário e gera os arquivos de aplicativos específicos do sistema operacional levando em consideração a camada do sistema operacional. Finalmente, o programa específico do sistema operacional será compilado usando o compilador suportado (por exemplo, ncc para TinyOS, gcc para Contiki). Para derivar o código binário de destino, os arquivos de criação são criados dependendo do hardware selecionado.

Figura 11 – Visão geral do processo de transformação e geração de código



Fonte: NGUYEN et al. (2015)

Em resumo, o FRASAD através de uma abordagem MDA, tendo como entrada um aplicativo genérico representado através de níveis elevados de abstração pode ser traduzido para diferentes plataformas especificas, conectando sensores para gerar aplicações da IoT. Dois estudos de caso foram fornecidos para mostrar a usabilidade e portabilidade de estrutura e os resultados demonstram que o FRASAD pode ser considerado como uma solução promissora

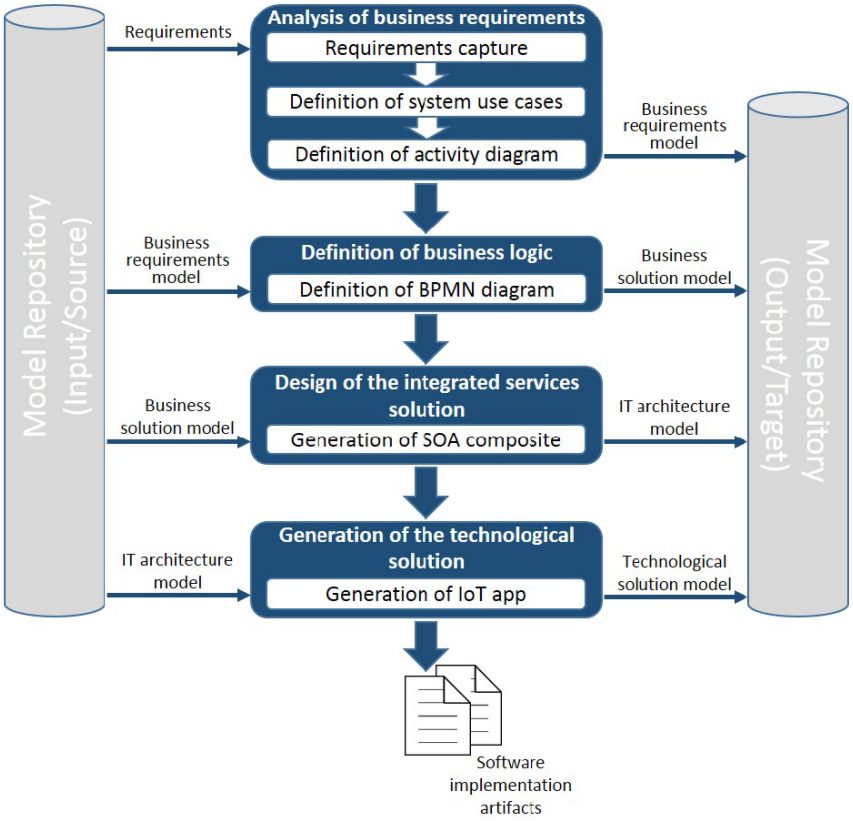
para reduzir a complexidade do desenvolvimento de *software* IoT.

O *framework* proposto pelos autores NGUYEN et al. (2015), apresentam uma estrutura para construir aplicativos de IoT para serem executados no sistema operacional do dispositivo, e não abordam integrações com serviços de computação em nuvem.

3.2.6 An approach based on model-driven development for IoT applications

SOSA-REYNA; TELLO-LEAL; LARA-ALABAZARES (2018) propuseram uma metodologia baseada em MDD composta de quatro fases com diferentes níveis de abstração, pontos de vista e granularidade. Os artefatos de saída das fases dessa metodologia são representados por modelos, gerados pela aplicação dos princípios do MDD. O resultado final são artefatos de implementação de *software*, ou seja, o código dos aplicativos ou sistemas de *software* para IoT. Na Figura 12, são apresentadas as fases que compõem esta metodologia: 1) análise dos requisitos de negócios, 2) definição da lógica de negócios, 3) projeto da solução de serviços integrados e 4) geração da solução tecnológica.

Figura 12 – Metodologia baseada em MDD para implantação de aplicativos de IoT



Fonte: SOSA-REYNA; TELLO-LEAL; LARA-ALABAZARES (2018)

- **Fase 1 - Análise dos requisitos de negócios:** o domínio do problema é analisado e os requisitos de negócios são identificados. Isso é feito considerando os requisitos funcionais

e não funcionais do sistema. Para definir o modelo de requisitos de negócios, é utilizada a linguagem UML, capturando o fluxo do processo de *software* por meio de diagramas de casos de uso e diagramas de atividades, gerando um modelo definido com alto nível de abstração independente de qualquer tecnologia ou linguagem de implementação.

- **Fase 2 - Definição da lógica de negócios:** essa fase enfoca o design dos processos necessários para suportar a lógica e os requisitos de negócios. Em seguida, o modelo de requisitos de negócios pré-gerado é usado como entrada para a fase e é complementado pela definição da lógica do processo de negócios, usando a linguagem BPMN, que permite gerar um modelo do negócio definido em um nível PIM do MDD, descrevendo o comportamento e as interações do processo de negócios de um ponto de vista global.

- **Fase 3 - Projeto da solução de serviços integrados:** um modelo da arquitetura de TI é definido, em um nível independente da plataforma para separar a solução de lógica de negócios dos aspectos técnicos de implementação, o que permite que esse tipo de implementação possa ser gerada em diferentes plataformas de destino. O modelo de arquitetura de TI é derivado do modelo da solução de negócios gerada na fase anterior, o modelo gerado permanece inalterado em qualquer plataforma. Nesse caso, o modelo de arquitetura de TI é gerado seguindo a abordagem orientada para os serviços SOA. A saída deste estágio é um modelo definido em um nível PIM.

- **Fase 4 - Geração da solução tecnológica:** nessa fase conceitos específicos da plataforma de implementação são usados para converter essa solução em um código executável de um aplicativo de *software* específico. Esta fase é realizada através da implementação de duas etapas: (1) projeto da solução específica da plataforma de TI, e (2) geração das especificações ou código do sistema de *software*. O primeiro estágio consiste na definição do modelo de especificações baseado em uma tecnologia padrão ou específica (por exemplo, TinyOS 2.0 ou WSN Operating Systems), usando como entrada para a fase, o modelo de arquitetura de TI, gerado anteriormente. A saída deste estágio é um modelo definido em um nível de PSM. O modelo de solução de tecnologia contém as informações necessárias para a plataforma específica (mensagens específicas no formato de envio ou recebimento de itens ou objetos, protocolos de transporte usados, UUID, remetente ou receptor do sensor). O segundo estágio consiste em uma transformação do PSM em texto, que representa o esqueleto do código ou o código executável de um aplicativo, geralmente em especificações baseadas em XML.

A metodologia é apoiada por métodos baseados na abordagem MDD, para reduzir custos e tempo de desenvolvimento, permitindo transformações de modelos automáticos e semi-automatizados para gerar os modelos de saída de cada fase. Uma transformação de modelo consiste em um conjunto de regras de transformação, que definem como um modelo de entrada é mapeado para um ou mais modelos ou códigos. Para suportar as transformações necessárias

do modelo para a metodologia, propõe-se a aplicação de diferentes métodos de transformação de modelos.

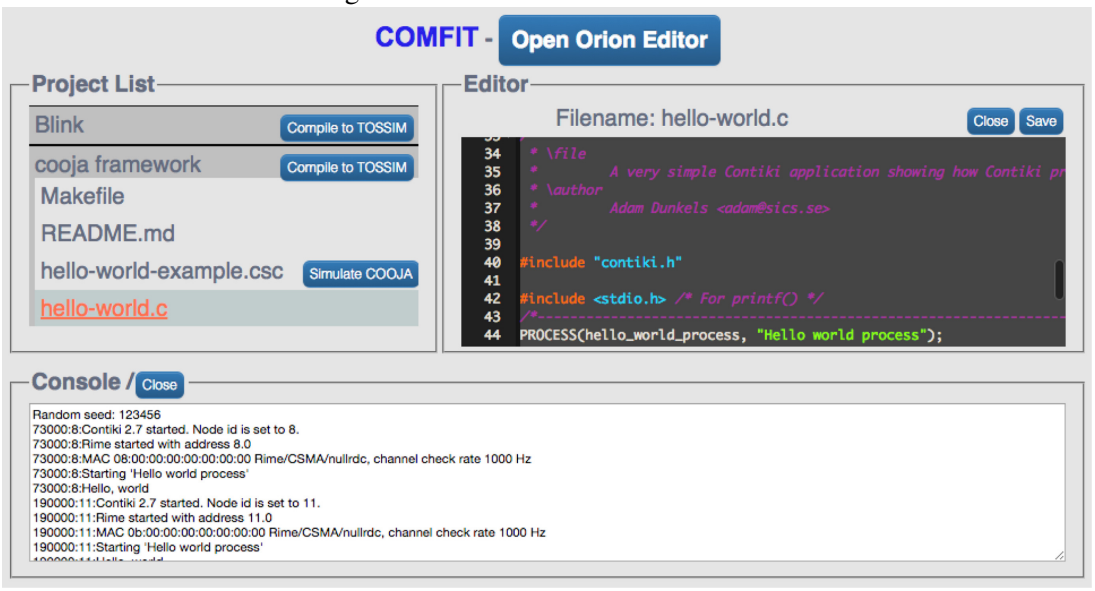
O objetivo da abordagem proposta é reduzir o tempo e os custos no desenvolvimento de *software*, implementando transformações de modelos automáticas e semi-automáticas. Além disso, foi proposta uma arquitetura para suportar os aplicativos ou sistemas de *software* para IoT. A arquitetura descreve, de forma genérica, as diferentes camadas necessárias para a implantação de aplicativos de software em IoT, utilizando os conceitos de Arquitetura Orientada a Serviços (SOA).

A metodologia proposta por SOSA-REYNA; TELLO-LEAL; LARA-ALABAZARES (2018) contempla o desenvolvimento de aplicações para o dispositivo, não prevendo os serviços de computação em nuvem.

3.2.7 COMFIT: A development environment for the Internet of Things

FARIAS et al. (2017) propuseram o COMFIT, um ambiente de desenvolvimento baseado em nuvem focado na criação de aplicativos para IoT. Seguindo uma arquitetura cliente-servidor e baseada na *Web*, o COMFIT engloba dois componentes principais: (i) computadores clientes e (ii) o servidor remoto na nuvem. No servidor remoto, os sistemas operacionais TinyOS e Contiki são hospedados e os respectivos ambientes de desenvolvimento são configurados para executar simulações e compilações. Assim, os desenvolvedores têm a opção de projetar aplicativos usando a infraestrutura MDA fornecida pelo COMFIT (instalado em suas máquinas clientes) ou usando uma interface *Web* fornecida pelo COMFIT para edição do código-fonte conforme exibido na Figura 13. Ao optar pelo design de aplicativos que usam a infraestrutura COMFIT MDA na máquina cliente, o usuário pode optar por carregar os arquivos de código-fonte para o servidor remoto para compilação ou simulação.

Figura 13 – Interface *web* do COMFIT

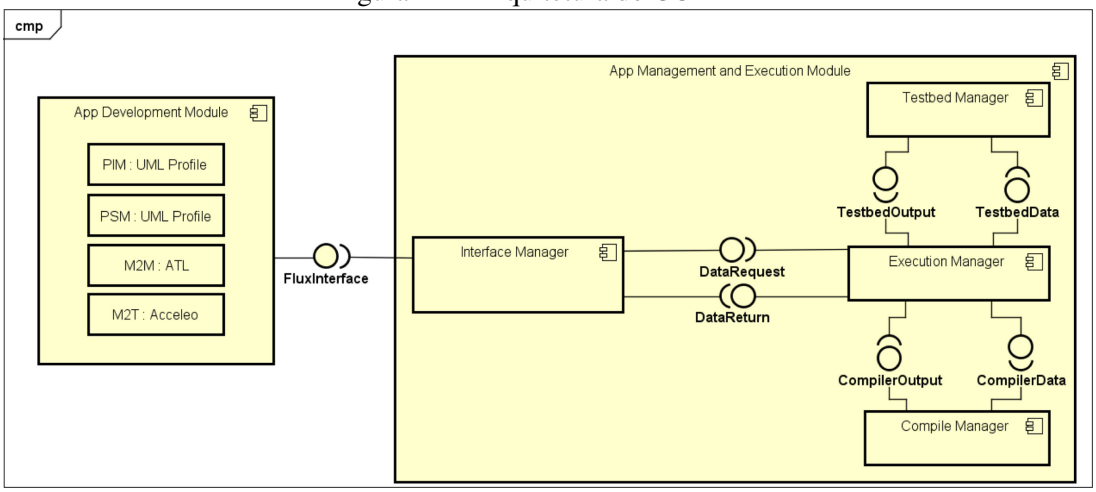


Fonte: FARIAS et al. (2017)

Composto por dois módulos diferentes: (1) o Módulo de Desenvolvimento de Aplicativos, uma infraestrutura de arquitetura orientada a modelos (MDA) e (2) o Módulo de Gerenciamento e Execução de Aplicativos, um módulo que contém interface da *Web* baseada em nuvem conectada a um servidor hospedado na nuvem com compiladores e simuladores para o desenvolvimento de aplicativos de IoT. O Módulo de Desenvolvimento de Aplicativos permite que os desenvolvedores projetem aplicativos de IoT usando modelos, que são adaptados à perspectiva do aplicativo ou à perspectiva da rede, criando assim uma separação entre essas duas preocupações. Esses modelos são automaticamente transformados em código através do Módulo de Desenvolvimento de Aplicativos, o COMFIT cria um ambiente onde não há necessidade de configurações adicionais para compilar ou simular adequadamente o código gerado, integrando o ciclo de vida de desenvolvimento de aplicativos IoT em um único ambiente parcialmente hospedado no lado do cliente e parcialmente na nuvem. COMFIT suporta geração automática de código, execução de simulações e compilação do código das aplicações com baixo esforço de desenvolvimento.

O Módulo de Gerenciamento e Execução de Aplicativos hospeda funções COMFIT que são executadas no servidor de nuvem e são compostas pelos componentes Gerenciador de Interface, Gerenciador de Execução, Gerenciador de Compilação e Gerenciador de Testes e os bancos de dados de plataforma e de teste, como pode ser visto na Figura 14.

Figura 14 – Arquitetura do COMFIT



Fonte: FARIAS et al. (2017)

Os autores FARIAS et al. (2017), propõem um ambiente de desenvolvimento baseado em nuvem focado na criação de aplicativos para IoT, no entanto essas aplicações são destinadas para o sistema operacional do sensor, portanto não contempla todas as características de uma aplicação CoT.

3.2.8 UML4IoT: A UML-based approach to exploit IoT in cyber-physical manufacturing systems

THRAMBOULIDIS; CHRISTOULAKIS (2016) apresentam um *framework* para enfrentar os desafios introduzidos pelo uso de IoT no processo de desenvolvimento de sistemas de manufatura. O UML4IoT permite que o desenvolvedor projete o componente ciber-físico usando os padrões de *software* e especificação de sistema amplamente aceitos, como a UML e SysML, e use um perfil UML para automatizar a transformação do componente com uma interface semelhante à IoT, pronta para ser integrada ao ambiente de fabricação de IoT. O perfil UML4IoT permite que o desenvolvedor gere automaticamente a interface compatível com IoT do componente ciber-físico e automatize o processo de desenvolvimento.

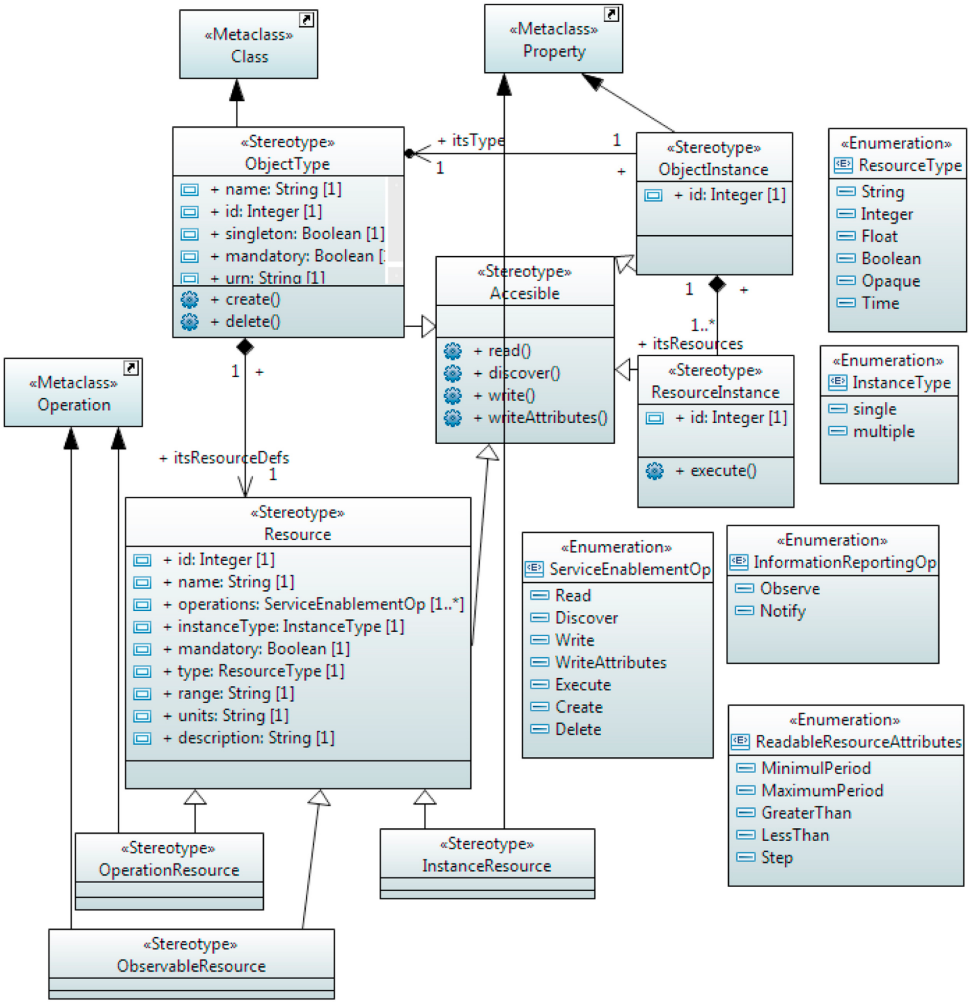
Para o caso de uma especificação de design UML não estar disponível, a abordagem pode ser aplicada no nível do código-fonte da parte cibernética. As propriedades do componente ciber-físico que devem ser expostas são anotadas corretamente e o código anotado resultante é usado para gerar automaticamente a camada que deve envolver o componente para apresentar uma interface compatível com IoT. Ambas as abordagens podem ser usadas para o desenvolvimento de novos componentes, bem como para a integração de componentes legados no moderno ambiente de fabricação de IoT.

A implementação do protótipo do sistema provou a eficácia da abordagem UML4IoT e, portanto, demonstrou sua aplicabilidade. Embora uma implementação parcial que suporta apenas

a interface de ativação de serviço tenha sido desenvolvida atualmente, a comparação de desempenho com outras implementações são uma indicação de que a abordagem é muito promissora, pois suporta uma geração totalmente automatizada com um pequeno custo no desempenho.

A Figura 15 apresenta o perfil UML proposto, que contém os principais elementos básicos necessários para expandi-lo e generalizá-lo, de modo a também serem aplicáveis em outros domínios. Como planos futuros incluem a implementação de outras interfaces, a implementação de um transformador para utilizar as anotações de tempo de edição para automatizar a geração da IoT, e a aplicação do perfil UML4IoT para outros domínios de aplicativos em que a IoT já foi adotada.

Figura 15 – A parte principal do perfil da UML4IoT



Fonte: THRAMBOULIDIS; CHRISTOULAKIS (2016)

Os autores THRAMBOULIDIS; CHRISTOULAKIS (2016), apresentaram um *framework* para lidar com a descoberta de novos dispositivos para apresentar uma interface de IoT para comunicação, portanto não contempla todas as características de uma aplicação CoT.

3.3 Análise Comparativa dos Trabalhos

Visando complementar os estudos dos trabalhos relacionados, nesta seção, será realizada a comparação dos trabalhos apresentados na Seção 3.2, juntamente com o trabalho desta pesquisa, com base em alguns critérios comparativos definidos a seguir. Para uma melhor compreensão, serão descritos brevemente os critérios utilizados para comparação dos trabalhos e o resultado dessa comparação será disponibilizada em um tabela comparativa relacionando os trabalhos com os critérios comparativos. O objetivo dessa comparação é identificar como esses critérios foram abordados nesses trabalhos, bem como identificar lacunas e apontar oportunidades de pesquisas. Os critérios de comparação são apresentados a seguir:

- **Abordagem MDD**: este critério tem como objetivo analisar os níveis de abstração alcançados pelos trabalhos nas etapas do processo de desenvolvimento de uma aplicação através de uma abordagem de desenvolvimento dirigido por modelos. As seguintes etapas foram analisadas:

  (a) Modelagem: verifica se a técnica analisada fornece algum recurso de modelagem da aplicação;

  (b) Configuração: se a abordagem prove um método para configuração de elementos da aplicação;

  (c) Transformação: avalia se o trabalho prove maneiras de realizar a transformação de modelos para modelos ou modelo para texto a fim de gerar o código fonte da aplicação;

  (d) Customização: verifica se o trabalho viabiliza a customização do código gerado;

  (e) Implantação: avalia se o trabalho prove uma abstração ao nível de implantação da aplicação para automatizar o processo.

- **Computação em Nuvem**: este critério avalia os trabalhos levando em consideração os requisitos para construir uma aplicação nativa para a nuvem, ou seja, que se beneficie dos recursos oferecidos pelos provedores de serviços em nuvem, tais como:

  (a) Auto-escalonamento: avalia se o trabalho atende em sua especificação ambientes auto-escalonáveis para execução da aplicação;

  (b) Provedor Agnóstico: verifica se o trabalho busca consumir serviços de nuvem indiferente do provedor a ser utilizado.

- **Funcionalidades de CoT**: este critério tem como objetivo comparar os trabalhos quanto algumas funcionalidades que contemplam os recursos para integração de componentes de IoT com os serviços de provedores de nuvem, tais como:

(a) Computação Cognitiva: avalia os trabalhos quanto a integração com serviços de computação cognitivas;

(b) Armazenamento: capacidade de armazenar as informações em serviços de nuvem;

(c) Processamento: capacidade de realizar o processamento das informações em ambientes de nuvem;

(d) Redes Sociais: integração com serviços de redes sociais e comunicação;

(e) *Publisher/Subscriber*: compatibilidade com protocolo de comunicação MQTT.

- **Geração da aplicação**: este critério tem como objetivo investigar os níveis de automação atingidos nos trabalhos para a geração de uma aplicação. Identificaram-se dois métodos:

  (a) Completa: isto é, aqueles trabalhos que alcançam a geração completa de uma aplicação;

  (b) Parcial: os trabalhos que geram apenas parte da aplicação.

- **Contexto de avaliação**: este critério verifica o contexto de avaliação ao qual o trabalho foi submetido, sendo classificada em:

  (a) Academia: avaliação realizada em ambiente acadêmico, com participação apenas de alunos e professores;

  (b) Indústria: avaliação realizada na indústria.

- **Métodos de avaliação**: este critério avalia os trabalhos quanto ao método de pesquisa podendo ser classificado como:

  (a) Pesquisa (*Survey*): trabalhos que envolvem entrevistas com os participantes;

  (b) Estudo de Caso: trabalhos que envolvem estudo minucioso com base em um cenário de uso real.

Com base nos critérios definidos, a Tabela 2 apresenta uma visão geral sobre a relação entre os trabalhos analisados e o trabalho apresentado nesta pesquisa com os critérios comparativos escolhidos nesta pesquisa, observa-se que todos os trabalhos contemplam algumas etapas do desenvolvimento da aplicação por técnicas de alto nível de abstração por abordagens dirigidas por modelos. Mas apenas os trabalhos de THRAMBOULIDIS; CHRISTOULAKIS (2016), FARIAS et al. (2017) e CAI et al. (2018) contemplam todos os critérios avaliados.

Tabela 2 – Comparativo dos Trabalhos Relacionados

Critérios		Trabalhos								
		JAISWAL et al. (2015)	NGUYEN et al. (2015)	PRAMUDIANTO et al. (2016)	THRAMBOULIDIS; CHRISTOULAKIS (2016)	FARIAS et al. (2017)	MEZGHANI; EXPOSITO; DRIRA (2017)	CAI et al. (2018)	SOSA-REYNA; TELLO-LEAL; LARA-ALABAZARES (2018)	MoT
Abordagem MDD	Modelagem	+	+	+	+	+	+	+	+	+
	Configuração	+	+	+	+	+	+	+	+	+
	Transformação	+	+	+	+	+	-	+	+	+
	Customização	-	+	+	+	+	-	+	+	+
	Implantação	-	-	-	+	+	-	+	-	+
Computação em Nuvem	Auto-escalonamento	*	*	-	*	*	+	+	*	+
	Provedor Agnóstico	*	*	-	*	*	-	+	*	+
Funcionalidades de CoT	Computação Cognitiva	*	*	-	*	*	-	*	*	+
	Armazenamento	*	*	-	*	*	-	*	*	+
	Processamento	*	*	-	*	*	-	*	*	+
	Redes Sociais	*	*	-	*	*	-	*	*	+
	Publisher/Subscriber	*	*	+	*	*	-	*	*	+
Geração da aplicação		P	P	P	P	C	P	P	P	C
Contexto de avaliação		*	A	A	I	A	A	A	*	A
Método de avaliação		*	E	E	E	E	E	E	*	E/S

Legenda:
(+) Contempla, (-) Não Contempla, (*) Não se aplica
(C) Completa, (P) Parcial
(A) Academia, (I) Indústria
(E) Estudo de Caso, (S) Survey

Fonte: Elaborado pelo autor.

Com relação aos requisitos de uso das funcionalidades de computação em nuvem, dos trabalhas analisados apenas um contemplou o requisito de prover uma abordagem agnóstica de provedores de serviços de nuvem CAI et al. (2018). Os autores propuseram uma estrutura da implementação de aplicativos móveis que mostrou-se mais adequada para implementar serviços em nuvem. Porém, este trabalho focou no desenvolvimento de aplicativos móveis baseados em nuvem, não se aprofundando em questões de IoT, o que aponta uma oportunidade de pesquisa em aberto. Propor maneiras para automatizar a integração dos componentes de IoT com serviços de computação em nuvem.

Sobre os requisitos de funcionalidades de CoT, que se refere à integração de IoT com provedores de computação em nuvem, nenhum dos trabalhos apresentou mecanismos para promover tais integrações de forma automatizada. O que reforça a oportunidade de propor melhorias para o desenvolvimento de aplicações CoT.

Quanto a geração da aplicação, todos os trabalhos apresentam uma geração parcial, exceto o trabalho de FARIAS et al. (2017), onde criam um ambiente onde não há necessidade de configurações adicionais para compilar ou simular adequadamente o código gerado. No entanto este trabalho, contempla o desenvolvimento de aplicações para serem executadas nos dispositivos de IoT.

Com relação ao contexto de avaliação, a maioria dos trabalhos foram avaliados no contexto acadêmico, somente o trabalho de THRAMBOULIDIS; CHRISTOULAKIS (2016) realiza uma avaliação com contexto industrial, os trabalhos de SOSA-REYNA; TELLO-LEAL; LARA-ALABAZARES (2018) e JAISWAL et al. (2015) não apresentaram uma avaliação de suas propostas. Em relação ao método de avaliação dos trabalhos, todos que realizaram avaliações utilizaram casos de estudos como alvo de suas propostas.

Concluindo a análise dos trabalhos selecionados, fica evidente que abordagens de desenvolvimento dirigidas por modelos estão sendo utilizadas para compor estudos nas áreas de IoT, e alguns trabalhos também promovem o desenvolvimento para a computação em nuvem, mas que nenhum dos trabalhos analisados contempla essas duas abordagens para promover a abstração dos problemas causados pela heterogeneidade dos diferentes provedores de computação em nuvem. Ampliar o uso de abordagens dirigidas por modelos para aplicações de CoT gera uma oportunidade de pesquisa.

Conforme apresentado na Tabela 2, a abordagem MoT, busca preencher as lacunas identificadas nos trabalhos analisados, contemplando a construção de uma aplicações CoT através de uma abordagem dirigida por modelos. Na próxima seção são apresentadas as oportunidades de pesquisa elencadas a partir da análise dos trabalhos relacionados.

3.4 Oportunidades de Pesquisa

Após a análise comparativa dos trabalhos relacionados apresentados neste capítulo, foram identificadas as oportunidades de pesquisa que podem contribuir ao processo de desenvolvi-

mento de aplicações CoT:

- **Abstração da heterogeneidade de ambientes de nuvem.** Com base nas análises realizadas, percebesse um esforço para abstrair a heterogeneidade presente nos dispositivos de IoT, mas não foram encontrados estudos que apresentam maneiras de abstrair também a heterogeneidade dos provedores de computação em nuvem. Sendo assim, seria uma oportunidade de pesquisa encontrar uma maneira de abstrair a complexidade dos serviços de computação em nuvem, para compor uma infraestrutura para execução de aplicações de IoT que fazem uso dos recursos de nuvem.

- **Abordagem de desenvolvimento para CoT.** Ao investigar os trabalhos apresentados fica evidente que não existe uma abordagem para o desenvolvimento de aplicações de CoT, e como foi constatado nos trabalhos relacionados, alguns esforços já estão sendo realizados com o desenvolvimento dirigido por modelos para IoT. Isso evidência como uma oportunidade de pesquisa a elaboração de uma abordagem que preencha as lacunas das atuais abordagens encontradas na literatura, as quais não contemplam importantes etapas do processo de desenvolvimento de aplicações CoT, incluindo modelagem, transformações, configuração e implantação.

- **Ferramenta de desenvolvimento integrada.** Com base nos trabalhos investigados, nenhum dos trabalhos apresentou uma ferramenta integrada para a criação de aplicações CoT. Ferramenta esta que contemple as principais etapas do processo de desenvolvimento de aplicações CoT. Portanto, esta dissertação trata o desenvolvimento de uma ferramenta de desenvolvimento que integre diferentes etapas do desenvolvimento de aplicações CoT como uma oportunidade de pesquisa.

- **Conhecimento empírico sobre CoT.** A tecnologia da informação está em constante evolução e novas tecnologias e abordagens surgem na academia e na indústria frequentemente. Realizar estudos empíricos com tais tecnologias seria uma oportunidade de produzir conhecimento técnico-científico, reduzindo a subjetividade do conhecimento encontrado na literatura atual. Além disso, o conhecimento gerado possibilitará identificar novas lacunas na literatura. Portanto, este trabalho trata a produção de conhecimento empírico sobre o desenvolvimento de aplicações CoT como uma oportunidade de pesquisa.

- **Avaliação na indústria.** Com base nas análises realizadas, apenas um dos trabalhos realizou uma avaliação na indústria da sua abordagem. Desse modo, há uma necessidade de estudos que explorem o desenvolvimento de aplicações CoT na indústria, bem como capturem a percepção de profissionais sobre os reais benefícios das técnicas utilizadas.

Dentre as oportunidades de pesquisa citadas, este trabalho busca destacar em seu desenvolvimento a elaboração de uma abordagem de desenvolvimento para CoT através da abstração

da heterogeneidade de ambientes de nuvem. Também propõe o desenvolvimento de uma ferramenta de desenvolvimento. No Capítulo 4 será apresentado a abordagem de desenvolvimento dirigida por modelos para aplicações CoT elaborada neste trabalho para abordar essas oportunidades de pesquisa.

4 MOT: UMA ABORDAGEM MDD PARA APLICAÇÕES COT

O objetivo central deste trabalho consiste em fornecer uma abordagem para o desenvolvimento de aplicações CoT que busca mitigar os problemas apontados no Capítulo 1. Sendo assim, neste capítulo é apresentada a MoT (Model of Things), uma abordagem de desenvolvimento de *software* dirigida por modelos para apoiar o desenvolvimento de aplicações de CoT, ou seja, aplicações de IoT que fazem uso da infraestrutura dos provedores de serviços de nuvem.

Este capítulo está organizado da seguinte maneira: A Seção 4.1 apresenta uma visão geral da abordagem. A Seção 4.2 detalha o perfil UML proposto. A Seção 4.3 detalha o processo de transformações dos modelos. A Seção 4.4 apresenta as principais características da abordagem. A Seção 4.5 apresenta a arquitetura da abordagem. Finalmente a Seção 4.6 detalha os aspectos para a implementação do MoT.

4.1 Visão Geral do MoT

A abordagem MoT busca auxiliar o desenvolvimento de aplicações CoT, neste contexto, a abordagem não trata dos componentes de IoT a nível de dispositivos físicos, ou seja, a abordagem conecta-se aos dispositivos a partir de um MQTT *broker* para então construir a aplicação CoT. A Figura 16 apresenta uma visão geral do processo de desenvolvimento de *software* proposto para dar suporte ao desenvolvimento de aplicações CoT. No total, o processo é composto por sete etapas: (1) Modelagem da Aplicação; (2) Transformação do Diagrama; (3) Configuração dos Componentes; (4) Transformação dos Modelos; (5) Implantação da Aplicação; (6) Customização dos Componentes e (7) Execução da Aplicação. A seguir são descritos os detalhes de cada uma destas etapas.

Figura 16 – Visão geral do processo de desenvolvimento.

Fonte: Elaborado pelo autor.

4.1.1 Etapa 1: Modelagem da Aplicação

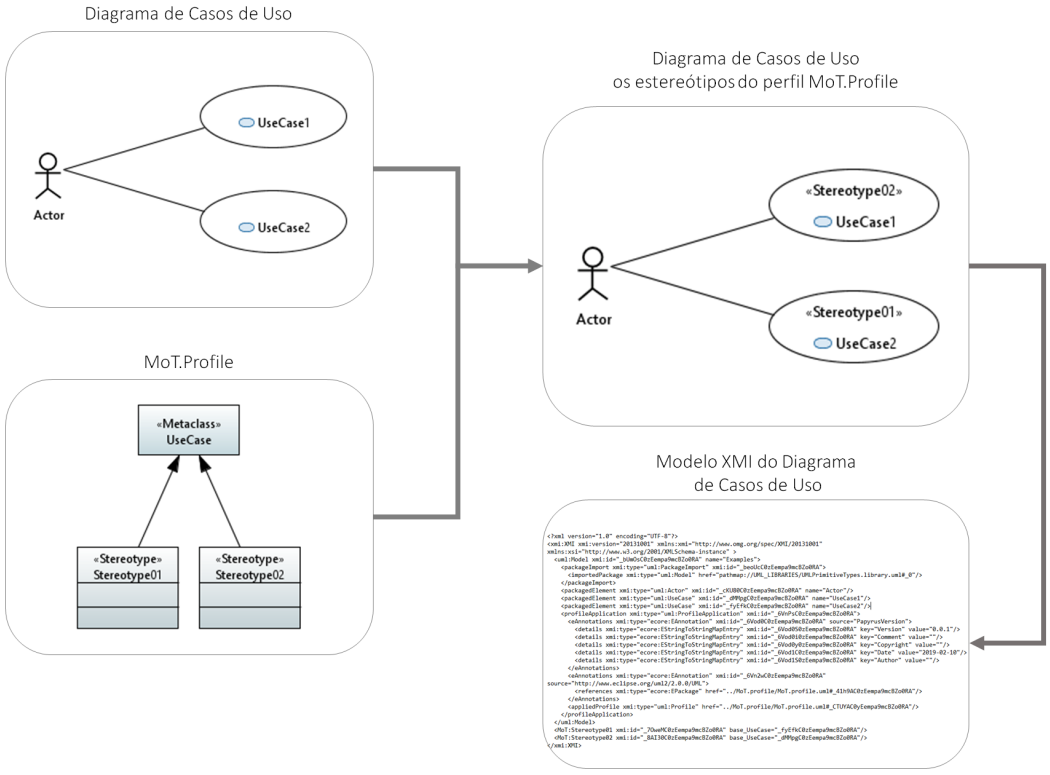
A etapa de Modelagem da Aplicação consiste na identificação e formalização dos requisitos, a fim de identificar as funções do sistema. É um processo iterativo, onde os requisitos são identificados e aperfeiçoados com discussões entre os especialistas de negócio e os especialistas de tecnologia. Após a definição dos requisitos da aplicação, eles são formalizados em modelos UML, como por exemplo, em um diagrama de casos de uso.

Através do diagrama de casos de uso, é possível representar as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários, sem se aprofundar em detalhes técnicos que dizem como o sistema será implementado.

O diagrama de casos de uso é utilizado como *input* para a abordagem junto com o perfil UML proposto denominado MoT.Profile, que será apresentado na Seção 4.2. A partir do perfil UML, é possível complementar as informações do diagrama de casos de uso, adicionando anotações semânticas. Essas informações adicionais ao diagrama não apresentam detalhes sobre a implementação, mas representam em alto nível de abstração as funcionalidades de uma aplicação CoT.

A Figura 17 apresenta o processo realizado na etapa de Modelagem da Aplicação. A partir dos elementos fornecidos pelo MoT.Profile, as funcionalidades presentes no diagrama de casos de uso são mapeadas para o domínio específico de aplicações de CoT.

Figura 17 – Processo da modelagem da aplicação.



Fonte: Elaborado pelo autor.

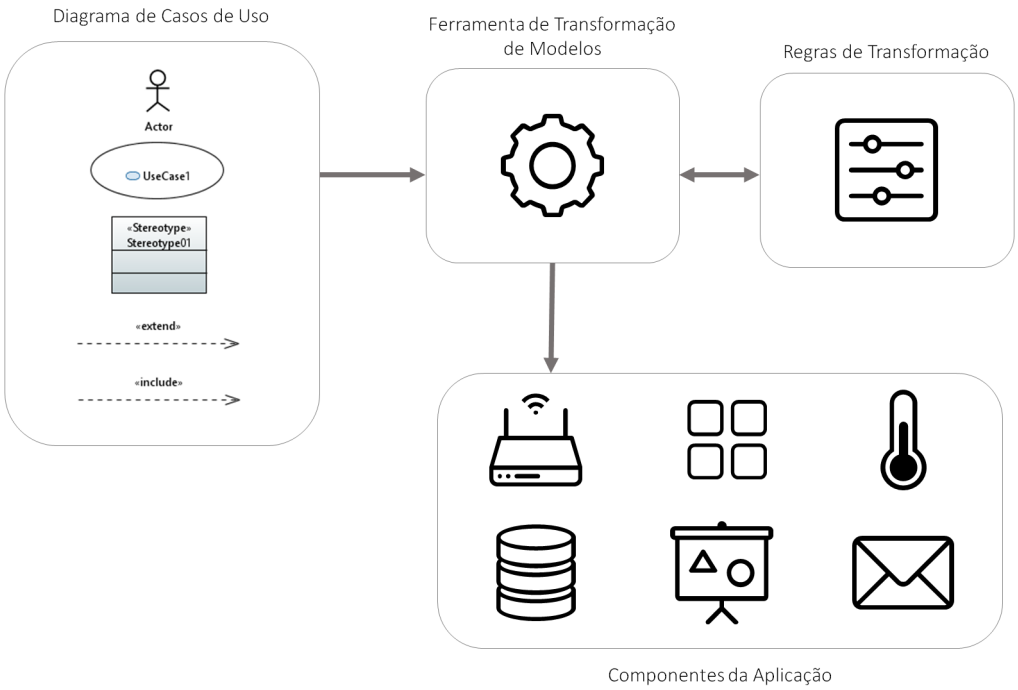
Com a extensão do diagrama através das funcionalidades do perfil UML, obtêm-se um modelo de casos de uso anotado com informações semânticas representadas através de estereótipos. Esse modelo então é representando textualmente através de um arquivo no formato XMI, que é um padrão estabelecido pelo OMG para troca de informações baseado em XML amplamente utilizado por ferramentas de modelagem UML, e será utilizado como *input* para a etapa seguinte, a Transformação do Diagrama.

4.1.2 Etapa 2: Transformação do Diagrama

Tendo o modelo da aplicação elaborado na etapa anterior representado em formato XMI, a etapa de Transformação do Diagrama compreende a primeira transformação de modelos proposta pela abordagem. Durante esse processo o transformador mapeia todos os elementos do diagrama e com a identificação dos estereótipos compatíveis com o perfil UML, realiza a transformação desses elementos UML em componentes da aplicação.

A Figura 18 apresenta o processo dessa transformação. Todos os elementos presentes no diagrama da casos de uso, elementos comuns do diagrama e os elementos do MoT.Profile utilizados no diagrama, são mapeados e analisados pela ferramenta de transformação de modelos, denominada de MoT.Transformer. Essa ferramenta contem um conjunto de regras de transformação, com as quais se baseia para gerar os componentes da aplicação, os detalhes dessa transformação são descritos na Seção 4.3.

Figura 18 – Processo de transformação do diagrama UML



Fonte: Elaborado pelo autor.

Os componentes da aplicação gerados a partir dessa transformação representam elementos e recursos específicos de uma aplicação CoT, e alguns deles necessitam de configurações adicionais não contempladas no diagrama de casos de uso. A próxima etapa da abordagem consiste na configuração desses componentes.

4.1.3 Etapa 3: Configuração dos Componentes

O processo de Configuração dos Componentes depende da interação do usuário para fornecer informações adicionais necessárias para determinado componente da aplicação. Essas informações são específicas para compor determinada aplicação como, por exemplo, informações para conexão a um banco de dados, para assinar/publicar informações de um sensor, informações de usuário e senha para redes sociais. Essas configurações são necessárias pois nem todas as informações necessárias são possíveis de representar através do diagrama UML utilizado.

Durante essa configuração, caso o usuário não possua dados de determinado serviço, como por exemplo, um serviço de banco de dados para armazenar informações referentes a aplicação, a abordagem MoT tem a capacidade de criar estes serviços em provedores de computação em nuvem. Dessa forma o processo para configuração de serviços de computação em nuvem não exige conhecimentos técnicos de plataformas desses serviços. A abstração de detalhes técnicos é uma das características do MoT. Acredita-se que a simplificação do processo de desenvolvimento de aplicações CoT passa pela abstração de detalhes de configurações dos serviços de computação em nuvem, componentes, incluindo componentes de *software*, dispositivos e sensores encontrados no contexto de IoT.

4.1.4 Etapa 4: Transformação dos Modelos

A próxima etapa é a geração automática de modelos específicos de plataforma através de uma abordagem generativa do MDE, que visa transformar modelo para modelo ou modelo para o código. A transformação dos modelos ocorre a partir das informações dos componentes da aplicação, onde a ferramenta MoT.Transformer traduz as informações dos componentes e suas configurações para componentes de uma plataforma específica.

Esses modelos representam o mesmo aplicativo modelado pelos interessados nas atividades anteriores, mas traduzem os detalhes de abstração de mais alto nível em detalhes de implementação específicos da plataforma, incluindo, por exemplo, tipos de conversão para os tamanhos específicos da plataforma, mapeando os elementos do aplicativo para a plataforma modelo de programação.

Além da transformação dos componentes para plataforma especifica, essa etapa também tem a responsabilidade de gerar o pacote de implantação, que contêm os arquivos de configuração da infraestrutura para a implantação e execução da aplicação em um ambiente de nuvem.

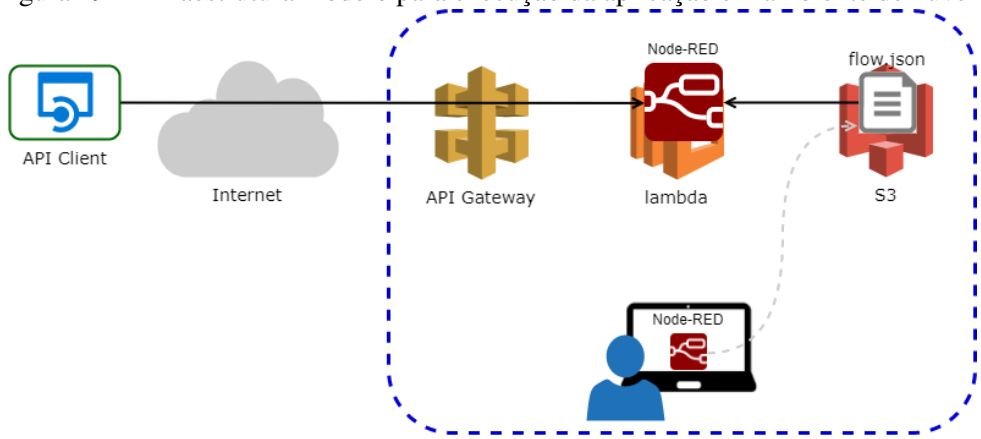
4.1.5 Etapa 5: Implantação da Aplicação

A etapa de implantação da aplicação, abrange a criação do ambiente de computação em nuvem para a execução da aplicação. Esse ambiente é criado com base no pacote de implantação gerado na etapa anterior. Com o objetivo de abstrair complexidade ocasionada pela heterogeneidade dos provedores de computação em nuvem, a abordagem visa automatizar a criação de um ambiente capaz de executar a aplicação. Indiferente do provedor de serviços de nuvem a implantação busca criar um ambiente para se beneficiar das vantagens da computação em nuvem.

A Figura 19 apresenta um modelo de infraestrutura criada para o provedor de computação em nuvem Amazon Web Services (AWS). Na Figura é possível identificar os serviços que compõem o ambiente da aplicação, sendo eles:

- **API Gateway:** atua como "porta de entrada"para que os aplicativos acessem dados, lógica de negócios ou funcionalidades em seus serviços *back-end*;
- **Lambda:** permite a execução de códigos sem provisionar ou gerenciar servidores;
- **S3:** serviço para armazenamento de arquivos.

Figura 19 – Infraestrutura modelo para execução da aplicação em ambiente de nuvem



Fonte: Elaborado pelo autor.

Além da criação do ambiente em nuvem, a etapa de implantação também realiza uma configuração para a execução da aplicação local, onde é possível realizar a customização dos componentes, a próxima etapa da abordagem.

4.1.6 Etapa 6: Customização dos Componentes

Em alguns casos, as transformações automáticas não abrangem todas as especificações para um componente da aplicação, por exemplo, informações de usuário e senha para serviços de

nuvem são informações delicadas e por esse motivo na etapa de configuração dos componentes não são informados. Portanto, a etapa de customização dos componentes é adequada para realizar as modificações específicas da plataforma que são necessárias após sua geração.

No entanto, é importante enfatizar que alguns componentes da plataforma gerados automaticamente, estarão prontos para uso. Alterações indevidas realizadas nessa etapa tendem a tornar o modelo UML original da aplicação obsoleto, o que desencoraja qualquer alteração na aplicação e caso o resultado obtido pela transformação não for o esperado, sugere-se a revisão da etapa de modelagem, pois o modelo UML fornecido pode estar inadequada ou incompatível com o perfil UML da abordagem MoT.

4.1.7 Etapa 7: Execução da Aplicação

Por fim, a última etapa da abordagem compreende então a execução da aplicação, podendo ser executada primeiramente em ambiente local para testes e homologação, e posteriormente a partir do ambiente de computação em nuvem criado na etapa de implantação.

4.2 Perfil UML

Pelo fato da UML ser uma linguagem de propósito geral que apresenta a capacidade de modelar um grande e diversificado conjunto de domínios de aplicações, em determinada situações não é capaz de modelar aplicações definidas em um domínio específico (por exemplo, área financeira, aplicações de tempo real, aplicações multimídia, entre outros). Sendo assim, surge a necessidade de criar uma extensão da linguagem. A UML pode ser estendida de duas maneiras: (a) com perfis para adaptá-la a plataformas específicas (por exemplo, J2EE / EJB) ou domínios (por exemplo, finanças e telecomunicações) ou (b) criando novos metamodelos (compostos de metaclasses e metarelationships) que modificam seu metamodelo.

A abordagem MoT propõe novos elementos de modelagem, trazendo conceitos específicos relacionados à aplicações de CoT, através do uso de um perfil UML. Perfis são definidos usando estereótipos, atributos e restrições. Estereótipos são o principal mecanismo em um perfil e permitem adaptar elementos para domínios específicos. Um estereótipo é um extensão de uma Metaclasse UML existente ou outros estereótipos, possivelmente definindo um conjunto de atributos adicionais.

A Tabela 3 apresenta alguns dos estereótipos do MoT.Profile, o perfil UML proposto pela abordagem MoT, e uma descrição da sua funcionalidade. Os estereótipos estão organizados por categoria de funcionalidade, como por exemplo, ≪SensorSubscribe≫ e ≪SensorPublish≫ pertencem à categoria de IoT pelo fato de suas funcionalidades representarem a conexão com um sensor de IoT.

Tabela 3 – Estereótipos do perfil UML do MoT

Categoria	Estereótipo	Descrição
IoT	≪SensorSubscribe≫	Monitora os dados de um sensor de IoT, conectando-se à um MQTT Broker e assinando as mensagens de um tópico específico e registrando os dados coletados em um banco de dados.
IoT	≪SensorPublish≫	Envia dados para um sensor de IoT, conectando-se à um MQTT Broker e publicando mensagens em um tópico específico.
Storage	≪DatabaseSave≫	Estabelece uma conexão à um servidor de Banco de Dados para salvar dados.
Dashboard	≪DashboardGauge≫	Apresenta informações em forma de indicadores em uma página de *Dashboard*.
Dashboard	≪DashboardChart≫	Apresenta informações em forma de gráfico em uma página de *Dashboard*.
Emotiv BCI	≪FacialExpression≫	Com a utilização do kit de ferramentas EmotivBCI permite criar integrações de interface cérebro-computador para capturar uma expressão facial.
Emotiv BCI	≪MentalCommand≫	Com a utilização do kit de ferramentas EmotivBCI permite criar integrações de interface cérebro-computador para capturar o valor de um comando mental de um perfil.
Social	≪TwitterPost≫	Conecta-se a uma conta do Twitter para realizar a publicação de um *post* na rede social.
Social	≪SendEmail≫	Conecta-se a um servidor de SMTP para enviar mensagens de e-mails.

Fonte: Elaborado pelo autor.

Esses estereótipos apresentados serão utilizados na validação da abordagem e são imprescindíveis para guiar o processo de transformação proposto pela abordagem e apresentado na Seção 4.3.

4.3 Transformações dos modelos

A abordagem dirigidas por modelos permitem uma abstração dos detalhes da tecnologia da plataforma de implementação e utilização de conceitos mais próximos do domínio do problema. Consequentemente, os modelos utilizados para desenvolver o sistema são mais fáceis de especificar, entender e manter, e são resistentes às alterações da tecnologia adotada para a implementação do sistema. Assim, o MDD promove a ideia de que através do foco nos modelos

pode-se obter melhor desenvolvimento e evolução do *software* (SELIC, 2003).

A especificação MDA coloca ênfase em um processo em camadas, usando diferentes pontos de vista. No MDA, um ponto de vista em um sistema é uma técnica que fornece uma maneira de representar funcionalidades de um sistema através de interfaces e padrões de design específicos, que caracterizam o comportamento e os processos de negócios de qualquer aplicativo implantado em uma plataforma sem se preocupar com detalhes técnicos.

O MDA propõe três tipos de pontos de vista: (i) Modelo Independente de Computação (CIM), (ii) Modelo Independente da Plataforma (PIM) e (iii) Modelo Específico da Plataforma (PSM). Esses modelos devem ser legíveis por máquina para que eles sejam sucessivamente transformados em esboços de código, esquemas, padrões de teste e scripts de implantação para diversas plataformas (KLEPPE et al., 2003).

Durante o processo da abordagem MoT, existem dois momentos onde ocorrem transformações de modelos. A etapa 2 Transformação do Diagrama e a etapa 4 Transformação dos Modelos:

- **Transformação do Diagrama:** ocorre a partir da análise dos elementos presentes no diagrama de casos de uso e dos estereótipos utilizados do perfil UML MoT.Profile. Para cada elemento e/ou estereótipo identificado nessa transformação, o MoT realiza um mapeamento específico desse elemento para um componente da aplicação. Esse mapeamento é definido pelas regras de transformação que estão detalhadas na Seção 4.3.1. A Figura 20 apresenta o algoritmo executado para a transformação do diagrama.

Figura 20 – Algoritmo de transformação do diagrama

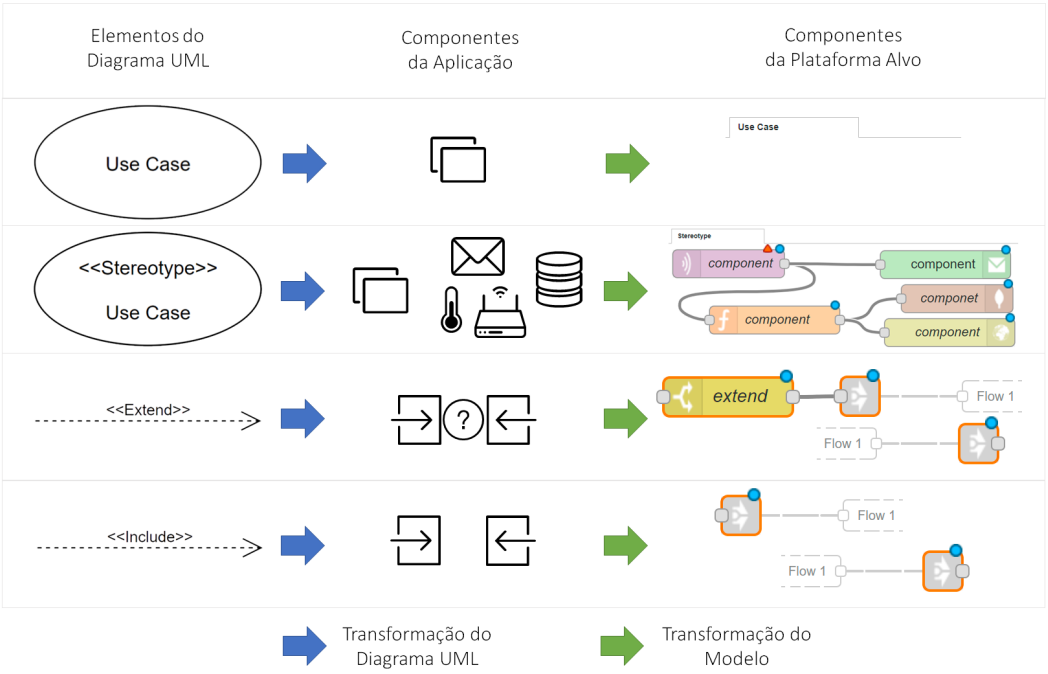
```
for all usecase in diagram do
  if usecase has stereotype then
    get stereotype definition
    save stereotype definition
    if stereotype has components then
      for all component in stereotype do
        get component definition
        save component definition
      end for
    end if
  end if
end for
```

Fonte: Elaborado pelo autor.

- **Transformação dos Modelos:** ocorre após as etapa de configuração dos componentes da aplicação. Nessa transformação esses componentes são transformados em componentes da plataforma alvo da aplicação.

A Figura 21 apresenta como os elementos do diagrama são mapeados para componentes da aplicação, e posteriormente para componentes da aplicação alvo.

Figura 21 – Mapeamento das transformações dos elementos do diagrama UML.



Fonte: Elaborado pelo autor.

- **Caso de Uso:** quando não estão fazendo uso de estereótipos para complementar sua funcionalidade, os casos de uso são mapeados para um componente da aplicação que representa um fluxo vazio, e da mesma forma para um componente da plataforma alvo. Esse componente não representa nenhum funcionalidade na aplicação, mas pode ser customizado na etapa 6 do processo apresentado na Seção 4.1.

- **Caso de Uso com estereótipo:** quando um caso de uso com estereótipo do Perfil UML é identificado pelo abordagem, o mesmo é mapeado para um conjunto de componentes que geram um fluxo definido pelas regras de transformações dos estereótipos que são descritas na Seção 4.3.1. Este fluxo é composto por outros componentes que tem a finalidade de atender ao propósito do estereótipo usado.

- ***Include*:** esse relacionamento quando identificado pela abordagem é mapeado para dois componentes específicos para criar o relacionamento entre os fluxos. Esses componentes são anexados aos fluxos que pertencem ao relacionamento e trabalham para realizar a conexão entre eles.

- ***Extend*:** esse relacionamento quando adiciona componentes para a conexão entre fluxos. O mapeamento do *Extend* se diferencia do *Include* por adicionar um componente extra

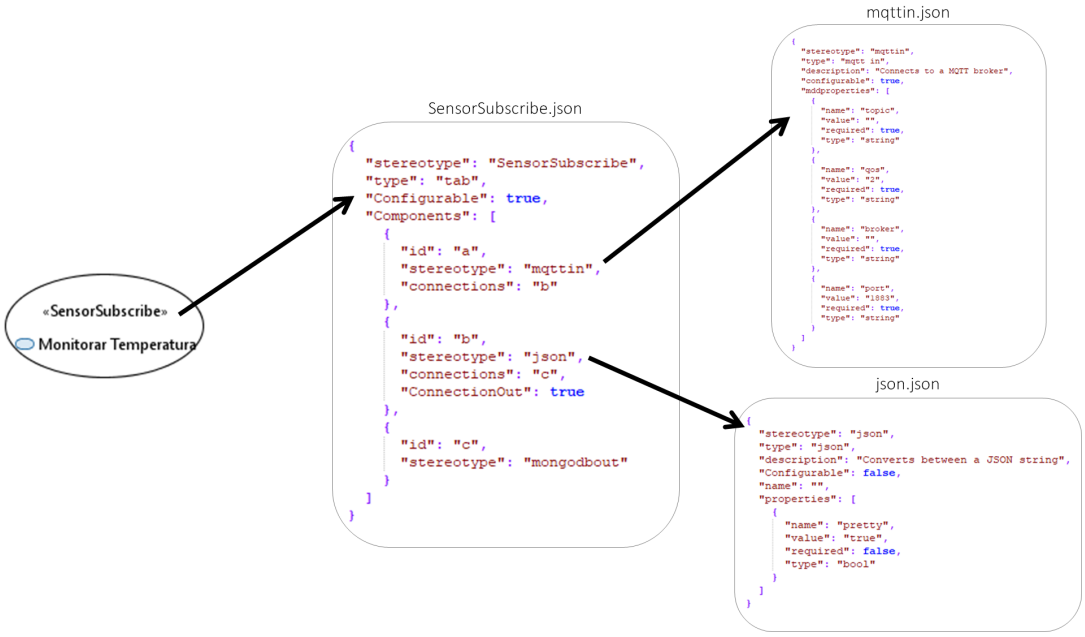
para criar uma cláusula de condição para identificar quando o fluxo principal da aplicação deve ser desviado para o outro fluxo alternativo.

4.3.1 Regras de Transformação

As regras de transformação guiam o processo de transformação do diagrama, onde ocorre o mapeamento dos elementos presentes no diagrama para componentes da aplicação. Durante esse processo, para cada estereótipo encontrado, o transformador busca em um repositório de arquivos, um arquivo no formato JSON que representa o estereótipo identificado. Ao identificar o arquivo do estereótipo, o transformador passa a interpretar o conteúdo do arquivo para identificar se ele representa um conjunto de componentes, ou seja, descobrir se o componente possui outros componentes que compõem a funcionalidade desse estereótipo.

Para cada componente encontrado dentro do arquivo o processo é repetido até que o transformador encontre o componente final, ou seja, que não possua outros componentes, mas sim propriedades. Essas propriedades são específicas do componente, e representam as configurações que o componente deve receber na etapa de Configuração dos Componentes. A Figura 22 apresenta o exemplo do mapeamento do estereótipo ≪SensorSubscribe≫, onde é possível identificar o processo de mapeamento dos estereótipos.

Figura 22 – Regras de transformações do estereótipo ≪SensorSubscribe≫



Fonte: Elaborado pelo autor.

Essas regras de transformações propõem que a abordagem seja extensível, pois para adicionar novas funcionalidades para a mesma, é necessário representar os componentes da aplicação em arquivos no formato JSON e posteriormente adicionar um estereotipo ao perfil UML apresentado na Seção 4.2.

4.4 Principais Características da MoT

Nesta seção, serão apresentadas as principais características que diferenciam a abordagem MoT de outros trabalhos relacionados ao desenvolvimento de aplicações CoT. Pode-se destacar como diferencial quatro características principais da abordagem:

- **Abstração dos componentes de IoT:** A abordagem proposta abstrai a complexidade dos componentes de IoT em modelos UML, o que permite que mesmo quem não possua conhecimento em programação consiga, por exemplo, desenvolver uma aplicação para conectar-se a sensores e/ou serviços oferecidos por redes sociais, provedores de computação em nuvem e APIs de outros provedores de serviços.

- **Geração automatizada do código-fonte:** Através de uma ferramenta de transformação de modelo para texto, a abordagem consegue gerar o código-fonte da aplicação com base nos modelos elaborados. Essa característica oferece grande produtividade para as equipes, pois possibilita o reuso dos modelos para construir diversas aplicações, e garante uma maior qualidade no *software* gerado.

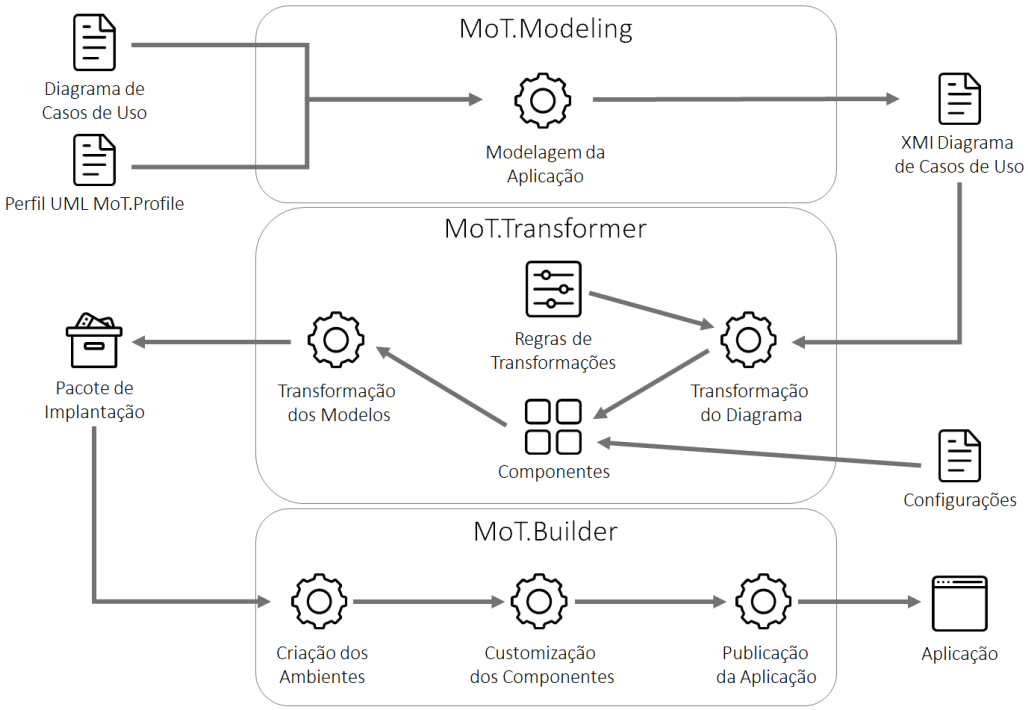
- **Configuração do ambiente:** Baseado nos componentes utilizados para construir a aplicação, a abordagem realiza a criação e configuração dos ambientes em provedores de computação em nuvem para dar suporte à aplicação CoT em desenvolvimento. Sendo assim, os detalhes de implantação são abstraídos pela abordagem o que reduz a complexidade de construir aplicações nativas da nuvem que aproveitam ao máximo os benefícios da computação em nuvem como, por exemplo, escalabilidade e pagamento por uso.

- **Extensibilidade:** Através da criação de novos estereótipos e definição das regras de transformação correspondentes é possível adicionar novas funcionalidades para a abordagem e com isso ampliar o uso da abordagem para outros domínios.

4.5 Arquitetura da Abordagem

Essa seção apresenta a arquitetura da abordagem MoT, essa arquitetura é composta por três módulos independentes, que encapsulam seu comportamento através de um conjunto de atividades internas, sendo eles: MoT.Modeling, MoT.Tranformer e MoT.Builder. A Figura 23 mostra uma representação da arquitetura e como seus principais módulos estão relacionados. A seguir, são apresentados os detalhes sobre cada módulo:

Figura 23 – Arquitetura da abordagem MoT



Fonte: Elaborado pelo autor.

- **MoT.Modeling:** Módulo responsável pela modelagem da aplicação. Recebe como entrada um diagrama de Casos de Uso, e as especificações do Perfil UML MoT.Profile. Para suporte a modelagem da aplicação existem ferramentas amplamente utilizadas na industria de *software* que são apresentadas na Seção 4.6. A premissa de utilização da ferramenta é o suporte à linguagem UML e sua extensão através do uso de perfis UML que constituem um mecanismo de extensão do padrão UML tradicional.

- **MoT.Transformer:** Módulo responsável pela análise, configuração e transformação dos modelos. Primeiramente ocorre a transformação do Diagrama baseado no conjunto de regras de transformações para identificar todos os componentes da aplicação, posteriormente esses componentes serão configurados com as informações necessárias. Por fim, ocorre a transformação dos modelos para então gerar como saída o pacote de implantação, que será tratado no próximo módulo.

- **MoT.Builder:** Módulo responsável pela criação dos ambientes para execução da aplicação, customização dos componentes e publicação da aplicação. A partir do pacote de implantação um conjunto de ferramentas é utilizado para automatizar a construção e publicação da aplicação.

Um conjunto de ferramentas é utilizado para a implementação da arquitetura apresentada.

Os detalhes de como cada módulo foi construído são apresentados na próxima seção.

4.6 Aspectos de Implementação

Essa seção apresenta os detalhes de implementação dos módulos que compõem a abordagem MoT. A seguir são apresentados as ferramentas utilizadas para a implementação de cada módulo da abordagem:

- **MoT.Profile:** o perfil UML da abordagem foi elaborado usando o Eclipse Papyrus[1], um ambiente de modelagem UML de software livre baseado no Eclipse que fornece suporte para a criação de perfis UML.

- **MoT.Modeling:** esse módulo visa a geração dos modelos da aplicação a ser criada. Essa modelagem pode ser realizada com ferramentas amplamente utilizadas na indústria de *software*, como Eclipse Papyrus[2], Astah[3], Modelio[4] entre outros disponíveis no mercado. A premissa de utilização da ferramenta é o suporte a linguagem UML e sua extensão através do uso de perfis UML que constituem um mecanismo de extensão do padrão UML tradicional. Também é imprescindível que seja possível exportar os modelos criados seguindo o padrão XMI, que é um padrão da OMG para troca de informações baseado em XML.

  As principais responsabilidades desta ferramenta são:

  - Modelagem da aplicação: através do uso da UML e do perfil UML da abordagem proposta é realizada a formalização dos requisitos em modelos que serão representados em um diagramas.
  - Exportação dos modelos: a partir dos modelos criados ocorre a exportação das informações contidas nos diagramas para arquivos XMI, que serão utilizados pelo módulo de transformação.

- **MoT.Transformer:** este módulo, tem a funcionalidade de realizar a gestão das aplicações criadas pela abordagem. Para tanto foi implementado uma ferramenta que tem como responsabilidade a interpretação dos arquivos XMI gerados pelos modelos UML na etapa de modelagem, bem como realizar a validação e as configurações dos componentes, e ao fim do processo de configuração realizar a transformação dessas informações para um arquivo de implantação que será utilizado pela ferramenta seguinte.

  Essa ferramenta consiste em uma aplicação *Web*, construída separando conteúdo (HTML), sua apresentação (CSS 3) e funções (JavaScript). Se fez uso da biblioteca de componentes

[1] https://www.eclipse.org/papyrus/
[2] https://www.eclipse.org/papyrus/
[3] https://astah.net/
[4] https://www.modelio.org/

de *front-end* mais popular, chamada Bootstrap, para criar projetos sensíveis e de primeira linha na *Web*. O Bootstrap proporciona interfaces amigáveis para dispositivos móveis, e responsivos para aplicações da *Web*. Para oferecer uma boa usabilidade para os usuários da abordagem, foram utilizados *plugins* criados no jQuery, que é uma biblioteca de JavaScript de código aberto que foi projetada para simplificar a execução de *scripts* do lado do cliente.

A ferramenta foi implementada usando a plataforma de desenvolvimento Visual Studio 2017 Community, a linguagem de programação C# e o ASP.NET CORE, que é uma estrutura de *software* livre, multiplataforma e de alto desempenho para a criação de aplicativos modernos conectados à *Internet* e baseados em nuvem. Para armazenamento das informações será usado um banco de dados Microsoft SQL Server 2018.

As principais responsabilidades desta ferramenta são:

- Interpretação do modelo UML: a ferramenta tem a capacidade de interpretar o arquivo XMI exportado pela ferramenta de modelagem UML, através das regras definidas pelo perfil UML da abordagem, e gerar os componentes pré-definidos da aplicação a ser criada.
- Configuração dos componentes: a partir dos componentes identificados na interpretação do modelo UML a ferramenta solicita ao usuário as informações necessárias e específicas de cada componente. Essas informações serão complementares ao modelo UML, esta configuração distingue uma aplicação de outra, pois o mesmo modelo UML pode ser usado para criar várias aplicações, mas essas configurações tornam a aplicação única.
- Transformação dos modelos: após a configuração de todos os componentes da aplicação a ser gerada, a ferramenta realiza a exportação das informações coletadas para um pacote de implantação, que contem as informações necessárias para a construção de aplicação pelo módulo MoT.Builder.

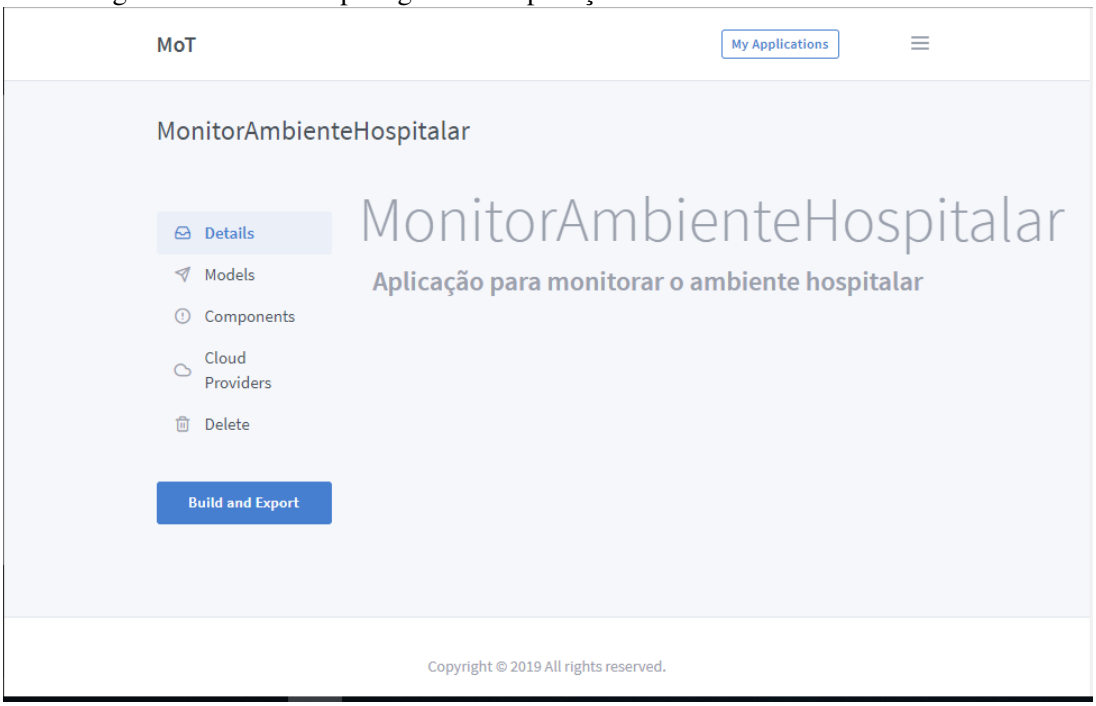
A Figura 24 apresenta a interface de gestão de aplicação da ferramenta implementada. Onde é possível identificar as opções para gerir uma aplicação CoT seguindo a abordagem, essas opções são detalhadas a seguir:

- Details: apresenta os detalhes da aplicação;
- Models: opção para importar os modelos criados na etapa de modelagem;
- Components: opção para acessar a listagem de componentes da aplicação e realizar a configuração deles;
- Cloud Providers: opção para configurar os dados de um provedor de serviços de computação em nuvem para a criação da infraestrutura para execução;

– Delete: opção para excluir a aplicação;

– Build and Export: opção para construir a aplicação e exportar o pacote de implantação.

Figura 24 – Interface para gestão de aplicação da ferramenta MoT.Transformar



Fonte: Elaborado pelo autor.

- **MoT.Builder:** o último modulo que compreende a execução da aplicação, customização dos componentes e por fim a publicação da mesma. Duas ferramentas se fazem necessárias nesta etapa, e ambas são executadas na plataforma Node.js, que permite escrever aplicações JavaScript para serem executadas no servidor (PEREIRA, 2014). A primeira ferramenta a ser usada é o NODE-RED[5], que é uma ferramenta de programação para conectar dispositivos de *hardware*, APIs e serviços online que fornece um editor baseado em navegador para facilitar a conexão de seus componentes para gerar um fluxo. Na interface do NODE-RED será exibido o fluxo da aplicação gerada pela abordagem proposta na etapa anterior, dessa forma o usuário pode executar a aplicação para identificar problemas na sua implementação e caso necessário, pode customizar os componentes gerados. A segunda ferramenta a ser usada nessa etapa mas que também executa sobre a mesma plataforma é o Serverless Framework[6], o qual é um kit de ferramentas para implantar arquiteturas sem servidor em qualquer provedor de hospedagem em nuvem que automatiza a configuração dos recursos da infraestrutura.

[5]https://nodered.org/
[6]https://serverless.com/

As principais responsabilidades destas ferramentas são:

- Execução da aplicação: Através da plataforma NODE-RED é possível executar a aplicação gerada, mesmo não estando em ambiente de produção, é possível identificar problemas no fluxo gerado pela abordagem, problemas de configurações e falhas na arquitetura da aplicação.
- Customização dos componentes: Caso necessário, através da interface do NODE-RED é possível editar os componentes da aplicação e suas configurações. Também pode-se estender as funcionalidades da aplicação, aplicado novos componentes, funções e outros serviços, porem essas alterações necessitam de algum conhecimento técnico na linguagem JavaScript e familiaridade com a plataforma.
- Publicação da aplicação: Após a verificação e validação das funcionalidades da aplicação, ela está pronta para ser publicada, através das ferramentas do Serverless Framework. O processo de publicação da aplicação é automatizado e após o processo é disponibilizado o endereço para acesso a aplicação gerada, considerando a aplicação entregue e em ambiente de produção.

5 AVALIAÇÃO

Este capítulo tem como objetivo apresentar a metodologia utilizada para avaliar a abordagem proposta no Capítulo 4 e discutir os resultados obtidos. A abordagem foi avaliada através da realização de um estudo de caso e um estudo experimental utilizando o modelo de aceitação de tecnologia. Essa avaliação faz parte dos objetivos específicos referentes a abordagem que foram listados na Seção 1.3, como também busca responder as questões de pesquisa apresentadas na Seção 1.2 deste trabalho.

Este capítulo esta organizado da seguinte maneira: A Seção 5.1 apresenta os detalhes referentes ao estudo de caso aplicado. A Seção 5.2 descreve o estudo experimental realizado utilizando o modelo de aceitação de tecnologia. Por fim, a Seção 5.3 apresenta as considerações finais referente as avaliações realizadas.

5.1 Estudo de Caso

Com o objetivo de validar a abordagem em relação a capacidade de gerar uma aplicação CoT, um cenário é fornecido para ilustrar o processo de desenvolvimento através da abordagem MoT. Desta forma é possível verificar a capacidade da MoT quanto a construção de uma aplicação CoT, que necessite do mínimo esforço manual para a geração do código-fonte e para configurações de ambientes em nuvem para dar suporte a execução da aplicação. A realização deste estudo de caso busca responder a primeira questão de pesquisa (QP-1) elaborada na Seção 1.2 deste trabalho.

5.1.1 Monitoramento de Ambiente Hospitalar

Ambientes hospitalares necessitam de monitoramento constante. Por se tratarem de ambientes críticos, onde pequenos problemas podem resultar em perdas de vidas humanas, existe uma série de regulamentações que disciplinam as condições nas quais tais ambientes devem estar para serem considerados aptos para acomodação e espera dos pacientes, bem como o armazenamento de vacinas e medicamentos. Tais ambientes necessitam de um correto acompanhamento da temperatura e umidade ao qual se encontram, em diversas unidades hospitalares este monitoramento é realizado de forma manual. Essa verificação manual apresenta diversos pontos falhos e uma maneira de resolver esses problemas é automatizar o processo de coleta dessas informações (CANTANHEDE; SILVA, 2014).

Como solução para o monitoramento de ambientes hospitalares, o autor CANTANHEDE; SILVA (2014) apresenta uma ferramenta baseada em IoT e computação autonômica para automatizar o processo de coleta de informações e sistematizar a forma como as informações coletadas podem ser analisadas e utilizadas para a tomada de decisões em resposta a possíveis situações adversas. Dentre os recursos oferecidos pelo sistema, podemos destacar análises a

partir da geração de gráficos e envio de alertas baseado em parâmetros customizados.

O cenário apresentado se concentra na coleta de informações de um sensor de IoT, armazenamento e o processamento desses dados para monitoramento e tomada de decisões. A partir desse cenários é possível definir alguns requisitos funcionais para uma aplicação, que são listados a seguir:

- Monitorar a temperatura de um ambiente;
- Armazenar as informações da temperatura;
- Gerar gráficos com as informações da temperatura;
- Enviar alertas quando a temperatura sofrer variação.

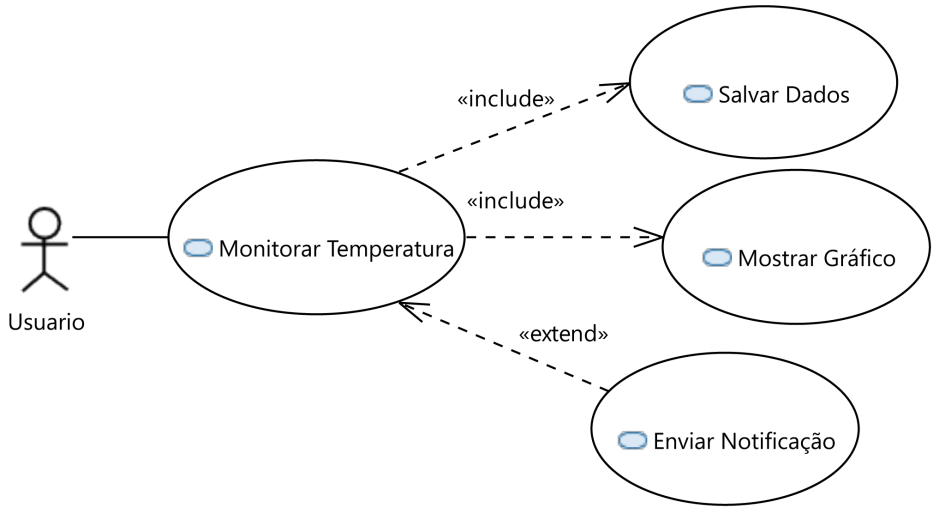
Com base nos requisitos extraídos do cenário apresentado, uma solução será implementada seguindo o processo da abordagem MoT apresentado no Capítulo 4, com o objetivo de produzir uma aplicação CoT que atenda aos requisitos identificados e ainda seja executada em ambientes de computação em nuvem. A seguir são apresentados os detalhes de implementação para cada etapa da abordagem.

Etapa 1: Modelagem da aplicação. O processo de desenvolvimento tem início com a etapa de modelagem da aplicação, onde os requisitos são identificados e formalizados em diagramas da UML, como por exemplo, em um diagrama de casos de uso. Para a modelagem da aplicação foi utilizada a ferramenta de código aberto Eclipse Papyrus[1], pois fornece um ambiente integrado para criação de diagramas utilizando linguagens de modelagem, entre elas a UML. Outra vantagem do Eclipse Papyrus é que ele também oferece um suporte a perfis UML e seus mecanismos de extensão.

A Figura 25 apresenta o diagrama de casos de uso elaborado com base no cenário apresentado anteriormente. No diagrama é possível identificar as funcionalidades do sistema sem se aprofundar em detalhes técnicos que dizem como o sistema será implementado.

[1] https://www.eclipse.org/papyrus

Figura 25 – Diagrama de casos de uso da aplicação

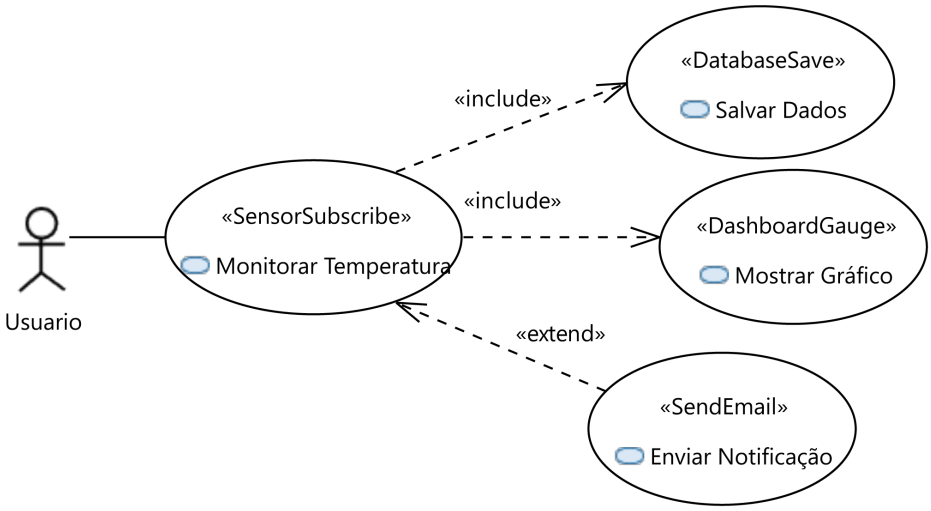


Fonte: Elaborado pelo autor.

A partir da importação do MoT.Profile, o perfil UML proposto pela abordagem e apresentado na Seção 4.2, na ferramenta de modelagem é possível complementar o diagrama de casos de uso, adicionando as anotações semânticas disponibilizadas pelo perfil UML. Essas informações adicionais ao diagrama não apresentam detalhes sobre a implementação, mas que representam em alto nível de abstração as funcionalidades de uma aplicação de CoT.

A Figura 26 apresenta o diagrama de casos de uso com as anotações do perfil UML, onde é possível identificar o uso dos seguintes estereótipos: ≪SensorSubscribe≫, ≪DatabaseSave≫, ≪DashboardGauge≫ e ≪SendEmail≫. Essas anotações guiarão os processos de transformação de modelos que serão realizados nas etapas seguintes.

Figura 26 – Diagrama de casos de uso da aplicação com os estereótipos do MoT.Profile



Fonte: Elaborado pelo autor.

Ao fim da elaboração do diagrama de casos de uso, com o uso dos recursos da própria ferramenta de modelagem UML, o diagrama é exportado para um documento XMI, que contem todos os elementos do diagrama de forma que possa ser interpretado por um transformador. A Figura 27 apresenta o arquivo resultante da exportação do diagrama elaborado, onde é possível identificar textualmente os elementos presentes no diagrama, bem como os estereótipos usados.

Figura 27 – XMI representando os elementos do diagrama de casos de uso

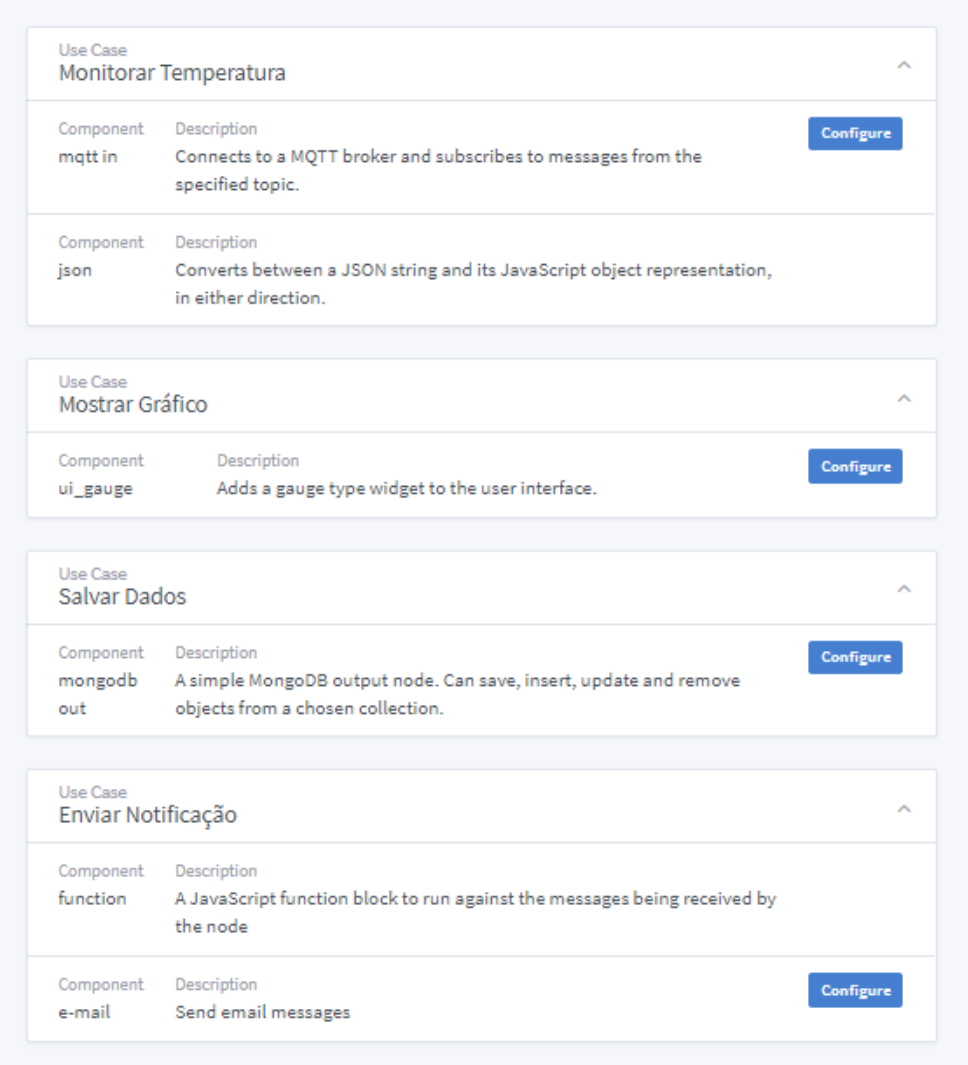
```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="20131001" xmlns:xmi="http://www.omg.org/spec/XMI/20131001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:MoT="https://cristianowelter.visualstudio.com/ModelOfThings" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:uml="http://www.eclipse.org/uml2/5.0.0/UML">
  <uml:Model xmi:id="_ERHn0BwjEemrk7v8KLXFVg" name="MonitorAmbienteHospitalar">
    <packagedElement xmi:type="uml:Actor" xmi:id="_GSitABwjEemrk7v8KLXFVg" name="Usuario"/>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_UyZo0BwjEemrk7v8KLXFVg" name="Monitorar Temperatura">
      <extensionPoint xmi:type="uml:ExtensionPoint" xmi:id="_ob9pYBwjEemrk7v8KLXFVg" name="ExtensionPoint2"/>
      <include xmi:type="uml:Include" xmi:id="_ngwlYBwjEemrk7v8KLXFVg" addition="_hYlGMBwjEemrk7v8KLXFVg"/>
      <include xmi:type="uml:Include" xmi:id="_d5eVUDlfEem-dNGzL9zigg" addition="_XW-4sDlfEem-dNGzL9zigg"/>
    </packagedElement>
    <packagedElement xmi:type="uml:Association" xmi:id="_eQLx4BwjEemrk7v8KLXFVg" memberEnd="_eQTGoBwjEemrk7v8KLXFVg
    _eQTGoRwjEemrk7v8KLXFVg">
      <ownedEnd xmi:type="uml:Property" xmi:id="_eQTGoBwjEemrk7v8KLXFVg" name="monitorar temperatura"
      type="_UyZo0BwjEemrk7v8KLXFVg"
      association="_eQLx4BwjEemrk7v8KLXFVg"/>
      <ownedEnd xmi:type="uml:Property" xmi:id="_eQTGoRwjEemrk7v8KLXFVg" name="usuario" type="_GSitABwjEemrk7v8KLXFVg"
      association="_eQLx4BwjEemrk7v8KLXFVg"/>
    </packagedElement>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_hYlGMBwjEemrk7v8KLXFVg" name="Mostrar Gráfico"/>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_kAlqQBwjEemrk7v8KLXFVg" name="Enviar Notificação">
      <extend xmi:type="uml:Extend" xmi:id="_ob8bQBwjEemrk7v8KLXFVg" extendedCase="_UyZo0BwjEemrk7v8KLXFVg"
      extensionLocation="_ob9pYBwjEemrk7v8KLXFVg"/>
    </packagedElement>
    <packagedElement xmi:type="uml:UseCase" xmi:id="_XW-4sDlfEem-dNGzL9zigg" name="Salvar Dados"/>
  </uml:Model>
  <MoT:SensorSubscribe xmi:id="_wJ4kIEnSEemRzMbvzg8ROA" base_UseCase="_UyZo0BwjEemrk7v8KLXFVg"/>
  <MoT:DatabaseSave xmi:id="_xUsKYEnSEemRzMbvzg8ROA" base_UseCase="_XW-4sDlfEem-dNGzL9zigg"/>
  <MoT:DashboardGauge xmi:id="_yj5IcEnSEemRzMbvzg8ROA" base_UseCase="_hYlGMBwjEemrk7v8KLXFVg"/>
  <MoT:SendEmail xmi:id="_zff1EEnSEemRzMbvzg8ROA" base_UseCase="_kAlqQBwjEemrk7v8KLXFVg"/>
</xmi:XMI>
```

Fonte: Elaborado pelo autor.

Concluindo a etapa de modelagem da aplicação, temos um modelo independente de plataforma para representar a as funcionalidades da aplicação. Este modelo é representado em formato XMI, ao qual será interpretado pelo módulo MoT.Tranformer na etapa seguinte do processo.

Etapa 2: Transformação do diagrama. A etapa de transformação do diagrama, tem como objetivo a criação da aplicação através do módulo MoT.Transformer. A partir da importação do arquivo XMI elaborado na etapa de modelagem, a ferramenta realiza a primeira transformação de modelos proposta pela abordagem, conforme apresentado na Seção 4.3. Durante esse processo o transformador mapeia todos os elementos do diagrama e com a identificação dos estereótipos compatíveis com o perfil UML apresentado na Seção 4.2, faz a transformação desses elementos em componentes da aplicação conforme apresentado na Seção 4.3. A Figura 28 apresenta a tela do protótipo onde são exibidos os componentes da aplicação, que são o resultado da transformação do arquivo XMI de entrada. Na Figura é possível identificar os casos de uso presentes no digrama e relacionados a eles, alguns componentes que foram criados para atender a funcionalidade proposta pelo estereótipo ao qual foi aplicado.

Figura 28 – Componentes da aplicação gerados para a partir da transformação do diagrama



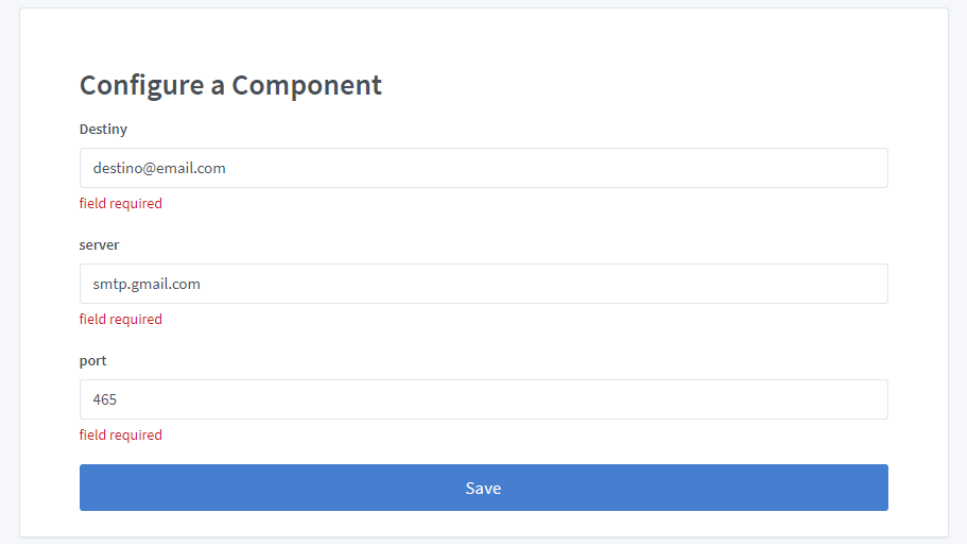
Fonte: Elaborado pelo autor.

Alguns componentes gerados a partir dessa etapa de transformação, ainda precisam de informações adicionais que devem ser inseridas pelo usuário. Dessa forma, a próxima etapa do

processo compreende a configuração desses componentes.

Etapa 3: Configuração dos Componentes Após a transformação do diagrama, a aplicação alvo apresenta componentes que são necessários para construir cada funcionalidade definida na modelagem da aplicação. Esses componentes variam de acordo com cada estereótipo usado e podem apresentar diversas funcionalidades, como por exemplo, funções, serviços de computação em nuvem, conectores à outros serviços, entre outros. Nem todos esses componentes estão prontos para uso, alguns deles precisam de informações complementares que devem ser informadas pelo usuário. O MoT.Transformer oferece recursos que ajudam o usuário a preencher essas informações. Conforme já apresentado na Figura 28, os componentes que precisam de configurações apresentam um botão que direciona o usuário para um formulário onde é solicitado as informações referentes ao componente. A Figura 29 apresenta as informações que devem ser fornecidas para o componente "e-mail", que contempla o caso de uso "Enviar Notificação".

Figura 29 – Formulário de configuração do componente e-mail



Fonte: Elaborado pelo autor.

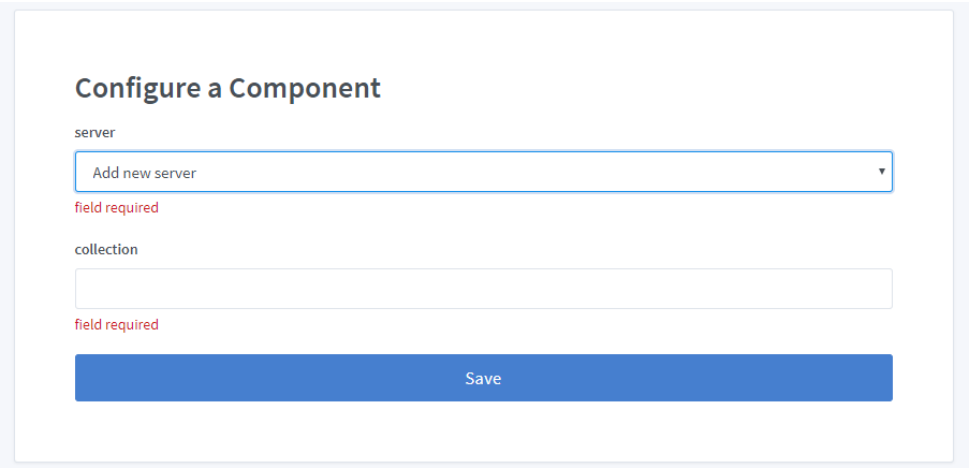
Conforme apresentado na Figura 29, um dos principais benefícios oferecidos pela abordagem MoT é a abstração da complexidade de especificações de linguagens de programação e plataformas de implementação. Dessa forma, a configuração de um recurso como o de envio de *e-mails* se resume ao usuário informar alguns campos em um formulário, sem se preocupar com os detalhes técnicos. Algumas informações sensíveis, como por exemplo, usuário e senha foram removidas da etapa de configuração do componente, sendo necessário ainda informar essas dados nas etapas seguintes.

Além da configuração de componentes, outro diferencial apresentado pela abordagem MoT é a capacidade de abstração da heterogeneidade das configuração de serviços dos provedores de computação em nuvem. Dessa forma através de algumas informações solicitadas no formulário

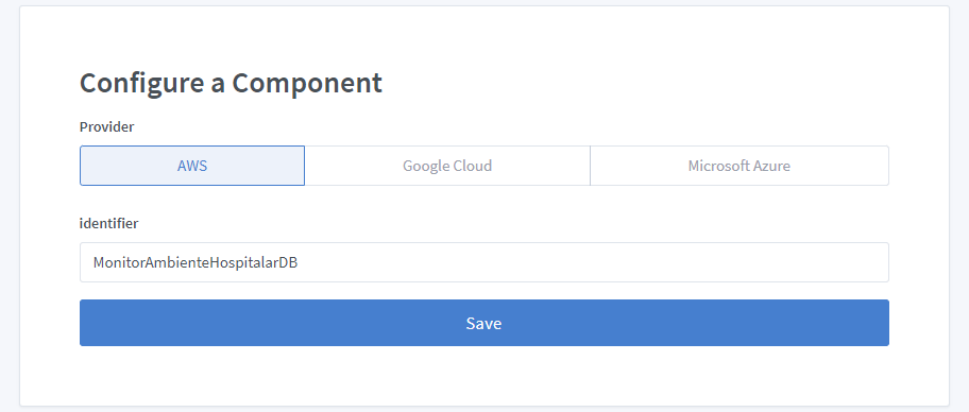
é possível instanciar um serviço necessário para a aplicação.

A Figura 30(a) apresenta a configuração do componente "mongodb out"que representa uma instância de banco de dados, este que é um serviço amplamente fornecido por provedores de computação em nuvem. Na configuração desse componente tem a opção de adicionar um novo servidor, que quando selecionado apresenta um novo formulário ilustrado na Figura 30(b). Esse formulário oferece ao usuários opções de escolha de um provedor de serviços em nuvem para a criação de uma nova instância do serviço. Ao selecionar um provedor, o módulo MoT.Transformer se encarrega de realizar as configurações necessárias para habilitar o serviço e configurar o componente da aplicação com os detalhes de conexão.

Figura 30 – Configuração do componente mongodb



(a) Campos de configuração do componente



(b) Criação de uma nova instância

Fonte: Elaborado pelo autor.

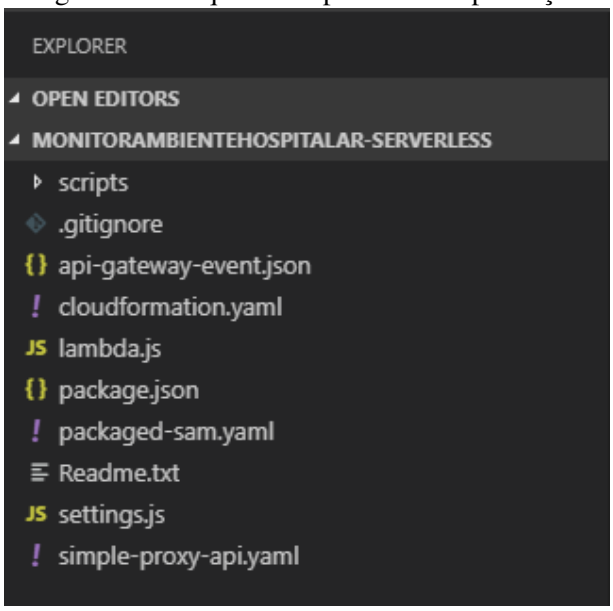
Após a conclusão das configurações de todos os componentes da aplicação, a próxima etapa do processo compreende a transformação desses componentes da aplicação para componentes de uma plataforma especifica.

Etapa 4: Transformação dos Modelos. A transformação dos modelos é a segunda transformação realizada pela abordagem MoT através do módulo MoT.Transformer, neste momento

ocorre a transformação dos componentes da aplicação e suas configurações para componentes da plataforma especifica. Conforme apresentado na Seção 4.6 a abordagem tem suporte para a geração de uma aplicação para a plataforma Node-RED, que por sua vez tem os recursos necessários para execução da aplicação.

Além da transformação dos componentes da plataforma especifica, essa etapa também tem a responsabilidade de gerar o pacote de implantação, que contêm os arquivos de configuração da infraestrutura para a implantação e execução da aplicação em um ambiente de nuvem. Conforme apresentado na Seção 4.6 esse pacote é composto por diversas ferramentas e *frameworks* que executam sobre a plataforma Node.js, e fazem parte do módulo MoT.Builder. A Figura 31 mostra a estrutura de arquivos de uma aplicação, que contêm desde arquivos de configuração, *scripts*, dependências de pacotes, entre outros.

Figura 31 – Arquivos do pacote de implantação



Fonte: Elaborado pelo autor.

Este pacote precisa ser baixado para o computador do usuário ao qual precisa ter a plataforma Node.js instalada para a implantação da aplicação.

Etapa 5: Implantação de Aplicação. A etapa de implantação da aplicação, é realizada pelo módulo MoT.Builder que é composto pelo ambiente de execução de Javascript Node.js, e alguns pacotes de dependência para compor a estrutura para a execução da aplicação, como por exemplo, o Node-RED, o Serverless Framework, entre outros. Todos os arquivos de configuração e as dependências estão presentes no pacote de implantação, que foi gerado na etapa de transformação dos modelos, conforme apresentado na Figura 31.

Para a implantação da aplicação é necessário a execução de dois comandos, que podem ser digitados no terminal do sistema operacional. O primeiro comando a ser executado é "npm run setup", ele tem a função de instalar todos os pacotes de dependências necessários para a

aplicação. Este comando também irá criar a configuração do ambiente modelo de computação em nuvem para execução da aplicação, conforme apresentado na Seção 4.6.

A Figura 32 apresenta a estrutura criada no ambiente da Amazon Web Services, onde é possível identificar as instâncias de serviços criados para suporte a aplicação. No topo, temos a função do Serviço AWS Lambda, um serviço para execução de código sem a necessidade de um servidor, esse serviço tem suporte para a execução de código da plataforma Node.js. Abaixo temos o serviço do Amazon S3, um serviço para armazenamento de arquivos, onde estão os arquivos de configuração da aplicação. O serviço API Gateway fornece URIs para acesso à execução da aplicação. E também, o serviço Amazon CloudWatch Logs que armazena registros das transações da aplicação.

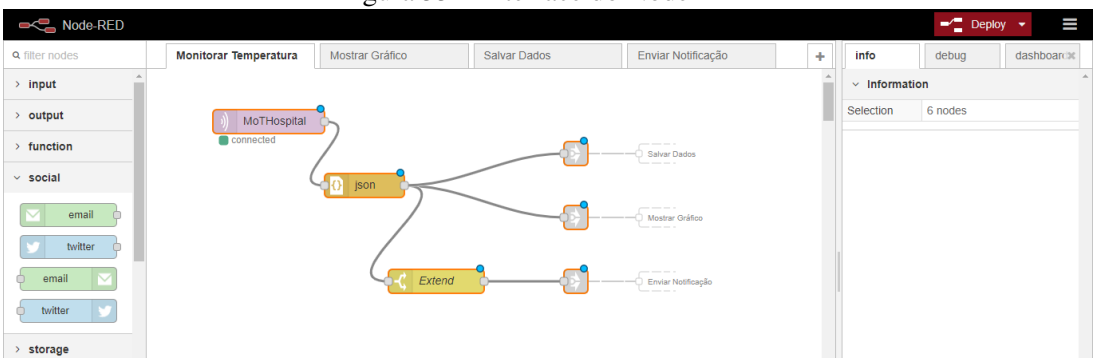
Figura 32 – Estrutura para execução da aplicação em ambiente de nuvem da AWS

Fonte: Elaborado pelo autor.

O segundo comando a ser executado nesta etapa do processo é o "node ./node_modules/node-red/red.js -s ./settings.js", esse comanda prepara um ambiente local para execução das configurações da plataforma Node-RED, que estará disponível para acesso através do endereço http://localhost:1880. Esse ambiente de execução disponibiliza a aplicação Node-RED para a etapa de customização dos componentes.

Etapa 6: Customização dos Componentes. A customização dos componentes é realizada através da própria interface do Node-RED, que pode ser acessada pelo endereço http://localhost:1880. A Figura 33 exibe a interface do Node-RED, onde é possível identificar que cada caso de uso presente no diagrama foi mapeada para uma aba especifica e o fluxo dos componentes representa a funcionalidade do estereótipo atribuído ao caso de uso. Na figura é apresentado logo ao centro o fluxo correspondente ao caso de uso Monitorar Temperatura, os elementos configurados e também os elementos que representam as ligações com os demais fluxos. A Figura 34 apresenta os detalhes correspondentes a cada caso de uso e os componentes que correspondem a cada funcionalidade.

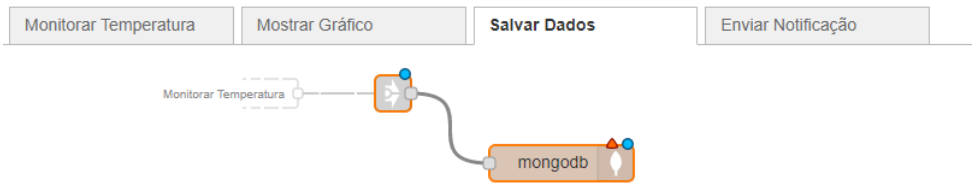
Figura 33 – Interface do Node-RED



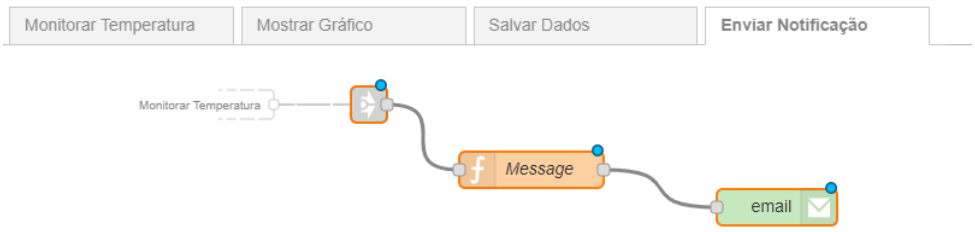
Fonte: Elaborado pelo autor.

Figura 34 – Fluxos dos componentes gerados no Node-RED

(a) Mostrar Gráfico



(b) Salvar Dados

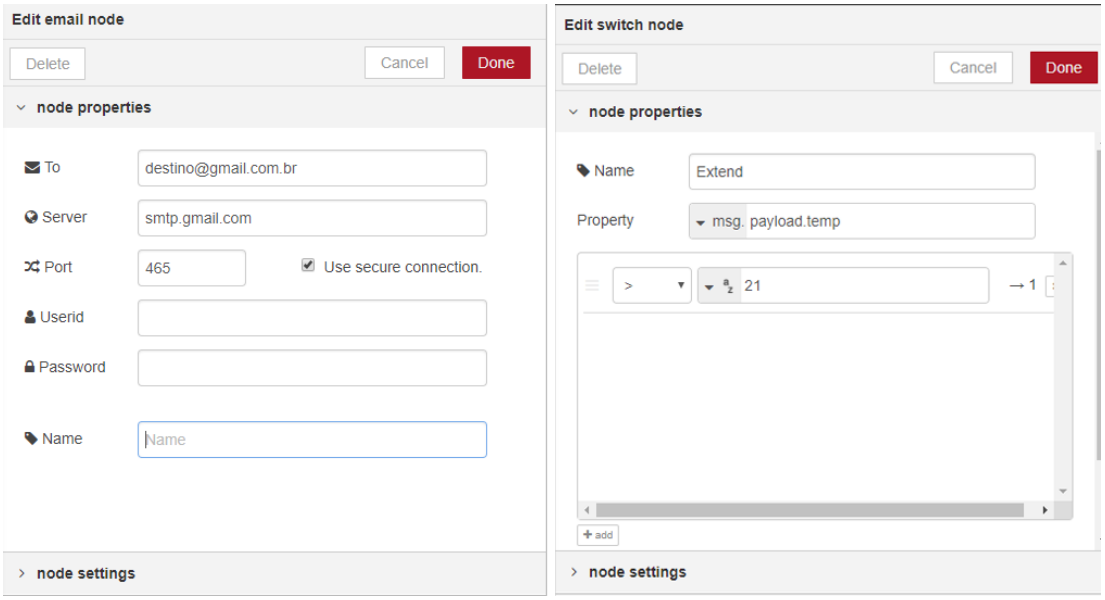


(c) Enviar Notificação

Fonte: Elaborado pelo autor.

Conforme mencionado na etapa de configuração de componentes, algumas informações sensíveis como usuários e senhas não foram configuradas na devida etapa. Para componentes que necessitam desses dados é necessário fazer esses ajustes nessa etapa. A Figura 35(a) apresenta o formulário padrão do Node-RED onde é possível informar os campos que não foram preenchidos na etapa de configuração. A Figura 35(b) apresenta os valores para configuração da cláusula de extensão do método para enviar a notificação caso a temperatura sofra alteração.

Figura 35 – Customização do componente e-mail



(a) Informações das credenciais do serviço (b) Configuração do valor para enviar a notificação

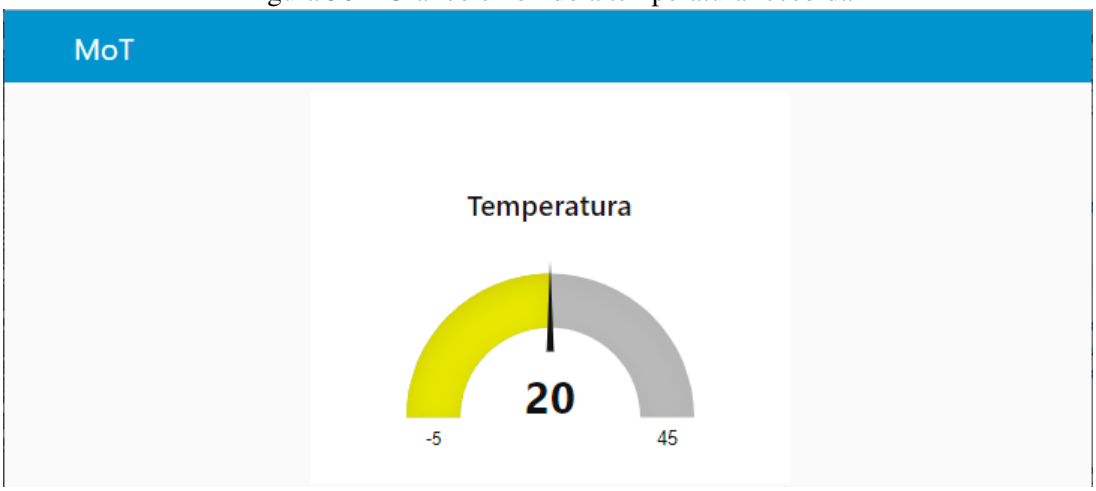
Fonte: Elaborado pelo autor.

Durante essa etapa, além das customizações realizadas nos componentes gerados pelo abordagem, o Node-RED oferece outros componentes que podem ser adicionados ao fluxo da aplicação, para criar novas funcionalidades para a aplicação que não estavam previstas pela abordagem. Ao fim da customização dos componentes, a aplicação está pronta para ser executada. O resultado é apresentado na ultima etapa da abordagem, a execução da aplicação.

Etapa 7: Execução da Aplicação. A última etapa da abordagem compreende então a execução da aplicação gerada seguindo o processo estabelecido. A execução se dá a partir do botão "Deploy"disponibilizado pelo Node-RED ao qual inicia a execução da aplicação local e atualiza os arquivos no ambiente criado na etapa de implantação de aplicação.

A Figura 36 apresenta o resultado final, após a instanciação da aplicação no Node-RED, onde é apresentada um indicador com a temperatura do ambiente a partir dos dados enviados pelo sensor. Além do indicador exibido, quando a temperatura sofrer uma variação conforme regra configurada na cláusula da extensão do fluxo de enviar notificação, uma mensagem de e-mail será enviada com os detalhes informados na etapa de configuração.

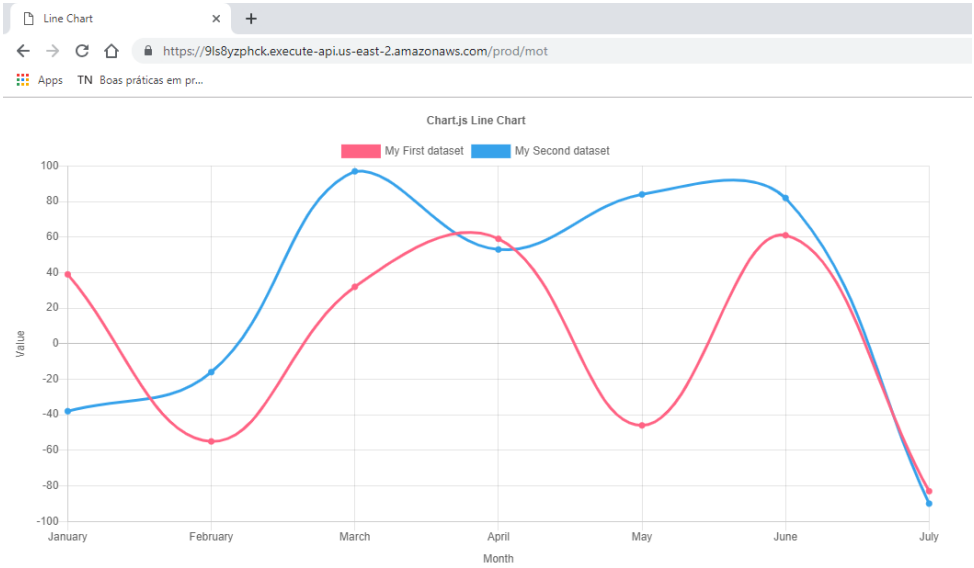
Figura 36 – Gráfico exibindo a temperatura recebida



Fonte: Elaborado pelo autor.

O gráfico apresentado na Figura 36 corresponde ao resultado da execução da aplicação em ambiente local. Para a execução da aplicação no ambiente criado no provedor de computação em nuvem é preciso acessar através das URIs disponibilizadas pelo provedor. Para a execução deste estudo de caso no ambiente de nuvem, foram encontradas inviabilidades técnicas que dizem respeito ao comportamento dos componentes do Node-RED, o que impossibilitou a execução dessa aplicação no ambiente de nuvem. Entretanto, conforme exibido na Figura 37 outros componentes do Node-RED executam corretamente no ambiente de nuvem criado para este cenário, o que reforça que a abordagem MoT tem potencial para construir uma aplicação CoT, mas ainda precisa de mais esforços para alcançar a geração completa da aplicação e do ambiente.

Figura 37 – Aplicação de gráfico estático, na plataforma Node-RED sendo executada no ambiente em nuvem



Fonte: Elaborado pelo autor.

5.2 Modelo de Aceitação de Tecnologia

Com o objetivo de avaliar a abordagem MoT quanto à percepção de usabilidade e aceitação pela perspectiva dos usuários, um estudo experimental é apresentado nesta seção seguindo o modelo de aceitação de tecnologia. Esta avaliação busca responder a segunda questão de pesquisa (QP-2) elaborada na Seção 1.2 deste trabalho.

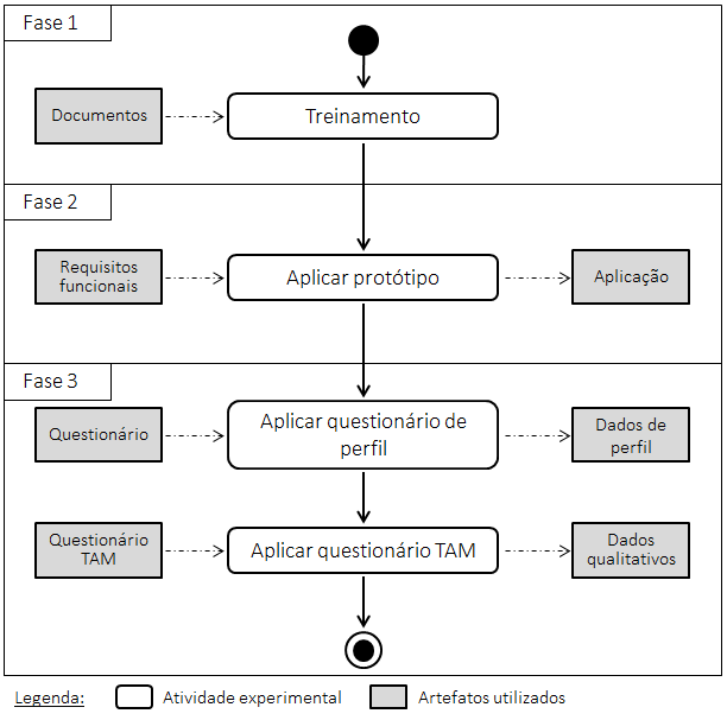
Esta seção esta organizado da seguinte maneira: Na Subseção 5.2.1 é apresentado o processo experimental utilizado. A Subseção 5.2.2 apresenta os questionários utilizados para a avaliação do MoT. A Subseção 5.2.3 descreve os cenários de avaliação. Na Subseção 5.2.4 define-se a seleção dos participantes. Por fim, a Subseção 5.2.5 apresenta os resultados obtidos.

5.2.1 Processo Experimental

A Figura 38 apresenta o processo experimental, formado por uma lista de atividades agrupadas em três fases descritas a seguir.

- **Fase 1:** primeiramente apresenta ao participante o objetivo do experimento e o que se espera alcançar com os testes. Nesta fase é realizado uma introdução ao conceito de aplicação CoT e também à abordagem MDD, posteriormente é realizado um treinamento de uso da ferramenta MoT.Transformer e MoT.Builder. Com isso é possível ajudar os usuários a entenderem melhor a utilidade do MoT e consequentemente verificar se entenderam a técnica proposta.

Figura 38 – Processo experimental



Fonte: Elaborado pelo autor.

- **Fase 2:** aplicar a abordagem MoT para a implementação do cenário de avaliação apresentado na Seção 5.2.3. Os participantes receberam os requisitos funcionais da aplicação (dados de entrada) e foram submetidos ao processo de desenvolvimento do MoT, contemplando todas as etapas da abordagem para a implementação da aplicação. Ao fim desta fase espera-se que os participantes tenham total entendimento da abordagem e, com isso, consigam avaliar o MoT na fase seguinte.

- **Fase 3:** possui duas atividades. A primeira tem o objetivo de coletar informações sobre o perfil dos participantes que irão responder uma lista de perguntas (dados de entrada) e as respectivas respostas são a saída dos dados coletados como resultado desta atividade.

  A segunda atividade tem o foco em aplicar o questionário de aceitação de tecnologia (TAM) como dados de entrada. Participantes recebem uma lista de perguntas sobre a percepção da facilidade de uso, percepção de utilidade e intenção de comportamento relacionados ao protótipo. Dados qualitativos (saída de dados) são gerados a partir da usabilidade e aceitação do protótipo, pela perspectiva dos usuários.

Os participantes realizam todas as atividades (Fase 1 a 3) para evitar qualquer inconsistência no processo experimental. Os dados coletados serão discutidos na próximo seção.

5.2.2 Questionários

Para responder as perguntas de pesquisa foram elaborados dois questionários para serem respondidos pelos participantes (introduzidos na subseção 6.4). Os questionários elaborados são detalhados a seguir.

Questionário 1: Perfil dos participantes. Tem como objetivo coletar dados referente as características e opiniões dos participantes. Criar um perfil dos participantes e analisar os dados coletados é importante para selecionar apenas usuários potenciais para os testes com o MoT. Para isso, perguntas foram feitas para coletar dados de características em geral, como idade, sexo, profissão, nível de escolaridade. Informações a respeito de experiências com tecnologias de IoT e computação em nuvem, além de coletar informações do tempo de experiência. Ao construir perfis de usuários, entende-se que é de suma importância para avaliar que a técnica proposta será experimentada por pessoas que tem o perfil de futuros usuários do MoT.

Tabela 4 – Perguntas de Caracterização do Participante

Categoria	Número	Pergunta
Caracterização do Participante	1	Qual a sua idade?
	2	Qual a sua profissão?
	3	Qual o seu grau de escolaridade?
Classificação de Experiência	4	Qual a sua experiência em análise de sistemas?
	5	Qual a sua experiência em desenvolvimento de *software*?
	6	Qual a sua experiência em Internet das Coisas?
	7	Qual a sua experiência em Computação em Nuvem?

Fonte: Elaborado pelo autor.

Questionário 2: Modelo de Aceitação de Tecnologia. Este segundo questionário avalia questões sobre a usabilidade e aceitação da abordagem e visa explorar a segunda questão de pesquisa (QP-2) apresentada na Seção 1.2 deste trabalho. Este questionário está baseado no Modelo de Aceitação de Tecnologia (TAM), proposto por DAVIS (1989) e revisado por MARANGUNIĆ; GRANIĆ (2015). O modelo TAM considera os seguintes itens como principais influências para a aceitação de uma nova tecnologia:

- **Facilidade de uso percebida:** grau em que uma pessoa acredita que a tecnologia poderia diminuir os seus esforços;
- **Utilidade percebida:** grau em que uma pessoa acredita que a tecnologia poderia melhorar o desempenho no desenvolvimento de suas atividades.

A pesquisa foi dividida em três categorias a fim de avaliar a aceitação do usuário em relação a aceitabilidade do MoT e foram desenvolvidas 8 afirmações que lidam com tópicos de

percepção de facilidade de uso, percepção de utilidade e intenção de comportamento. conforme a Tabela .

Tabela 5 – Perguntas Sobre Utilidades e Facilidades da Ferramenta

Categoria	Número	Afirmação
Utilidade do Sistema	8	A abordagem MoT facilitaria o desenvolvimento de aplicações de CoT
	9	A abordagem MoT ajudaria na produtividade dos desenvolvedores
	10	A abordagem MoT reduziria o tempo para entregar uma aplicação de CoT
	11	A abstração dos componentes de IoT reduz a complexidade do desenvolvimento
	12	A abstração dos componentes de Computação em Nuvem reduz a complexidade da configuração dos ambientes
Facilidade de Uso	13	O protótipo do MoT fácil de usar
	14	A abordagem do MoT é fácil de aprender
	15	A abordagem do MoT é fácil de dominar
Percepção Geral	16	A abordagem MoT poderia ser usada para desenvolvimento de aplicações de CoT na indústria de software
	17	Uma pessoa com pouco conhecimento técnico em desenvolvimento de IoT e computação em nuvem poderia construir uma aplicação de CoT doméstica
	18	Eu usaria a abordagem do MoT para construir minha aplicação de CoT

Fonte: Elaborado pelo autor.

Usando a escala *Likert* (LIKERT, 1932) de cinco pontos, os participantes podem classificar as afirmações variando entre 1 (discordo totalmente) até 5 (concordo totalmente). Ao final do questionário foi disponibilizado um espaço livre para comentários, criticas e elogios.

5.2.3 Cenários de Avaliação

Com o objetivo de avaliar a abordagem proposta, foram criados 4 cenários. Estes cenários representam etapas do processo da abordagem e são descritos abaixo:

- **Modelagem da aplicação:** o primeiro cenário contempla a modelagem da aplicação, compreende o processo de estender o diagrama de casos de uso tradicional com os estereótipos fornecidos pelo perfil UML do MoT, através de uma ferramenta de modelagem UML convencional. Neste cenário é solicitado ao usuário para adicionar os estereótipos correspondentes a cada funcionalidade em seu respectivo caso de uso do diagrama.

- **Transformação do diagrama e configurações dos componentes:** este segundo cenário possibilita ao usuário utilizar o protótipo da ferramenta MoT.Transformer para importar o XMI do diagrama e posteriormente realizar as configurações dos componentes da aplicação. O usuário deve importar o arquivo XMI da representação do diagrama de casos de uso para gerar os componentes da aplicação na ferramenta e concluir as configurações desses componentes através do formulário de configuração de cada componente.

- **Transformação dos modelos e implantação da aplicação:** neste cenário os usuários podem então através da ferramenta MoT.Builder, realizar a implantação da aplicação para então construir sua aplicação de CoT. O usuário faz a exportação do pacote de implantação na ferramenta, onde a partir do pacote, precisa efetuar os comandos para a geração dos *scripts* que irão construir os ambientes para a execução da aplicação.

- **Customização dos componentes:** este cenário apresenta maior complexidade pois envolve conhecimentos técnicos específicos da plataforma para customizar os componentes da aplicação. O usuário é solicitado a customizar o componente *Extend* para atribuir um valor para direcionar o fluxo da aplicação para o novo fluxo.

5.2.4 Seleção dos Participantes

Os participantes da avaliação foram selecionados por conveniência entre alunos de universidades da região com formação em áreas de tecnologia da informação e profissionais da área de tecnologia, todos com alguma experiência em desenvolvimento de *software*. Assim foi possível obter participantes com diferentes perfis de conhecimento e experiências.

No total 12 participantes realizaram o processo de avaliação do MoT. Dentre esses participantes, 10 (83,3%) são do sexo masculino e 2 (16,7%) são do sexo feminino. A faixa etária dos participantes ficou entre 25 a 40 anos. Com relação ao nível de formação todos possuem no mínimo graduação completa em cursos de Tecnologia. Dos participantes 11 (91,7%) atuam em empresas de tecnologia e 1 (8,3%) é estudante. Os detalhes dos perfis dos participantes é apresentado a seguir.

Dos participantes selecionados 6 tem idade entre 25 a 29 anos, 2 entre 30 a 34 anos, 3 entre 35 a 39 anos e apenas 1 entre 40 a 44 anos. A Tabela 6 ilustra a faixa etária dos participantes.

Tabela 6 – Faixa etária dos participantes

Faixa etária	**Participantes**
Entre 25 a 29 anos	6
Entre 30 a 34 anos	2
Entre 35 a 39 anos	3
Entre 40 a 44 anos	1

Fonte: Elaborado pelo autor.

Referente ao nível de escolaridade dos participantes 9 possuem nível de graduação na área

de tecnologia e 3 possuem nível de mestrado, conforme detalhado na Tabela 6, que representa o nível de escolaridade dos participantes.

Tabela 7 – Nível de escolaridade dos participantes

Nível de escolaridade	**Participantes**
Graduação	9
Mestrado	3

Fonte: Elaborado pelo autor.

Dos 12 participantes da avaliação 1 é estudante e 11 ocupam empregos profissionais na área de tecnologia, sendo que 6 atuam como programadores/desenvolvedores de sistemas, 4 atuam como analistas de sistemas e 1 atua como engenheiro de *software*, conforme descrito na Tabela 8.

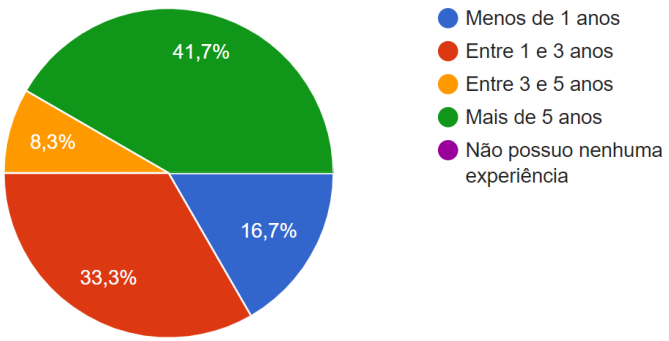
Tabela 8 – Profissão dos participantes

Profissão	**Participantes**
Programador	6
Analista de Sistemas	4
Engenheiro de Software	1
Estudante	1

Fonte: Elaborado pelo autor.

Referente à experiência dos participantes em análise de sistemas, 5 participantes possuem mais de 5 anos de experiência, 1 possui entre 3 e 5 anos, 3 possuem entre 1 e 3 anos e 2 participantes possuem menos de 1 ano de experiência. Na Figura 39 é ilustrado o percentual dos participantes por tempo de experiência em análise de sistemas.

Figura 39 – Tempo de experiência em análise de *software* dos participantes

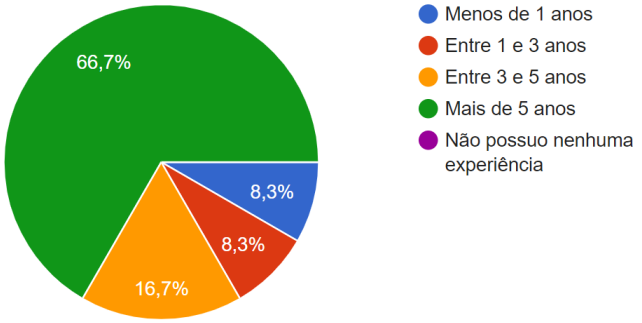


Fonte: Elaborado pelo autor.

Sobre a experiência dos participantes em desenvolvimento de *software*, 8 participantes possuem mais de 5 anos de experiência, 2 possuem entre 3 e 5 anos, 1 possui entre 1 e 3 anos e 1

participante possui menos de 1 ano de experiência. Na Figura 40 é ilustrado o percentual dos participantes por tempo de experiência em desenvolvimento de *software*.

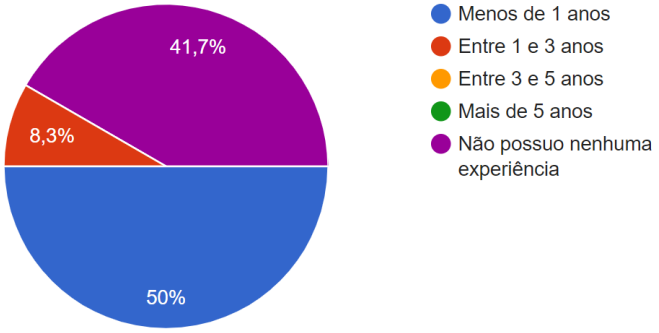
Figura 40 – Tempo de experiência em desenvolvimento de *software* dos participantes



Fonte: Elaborado pelo autor.

Com relação a experiência dos participantes em IoT, 5 participantes não possuem nenhuma experiência e 6 participantes possuem menos de 1 ano e apenas 1 possui entre 1 e 3 anos de experiência. Abaixo na Figura 41 é ilustrado os dados referentes aos participantes no que diz respeito a experiência em IoT.

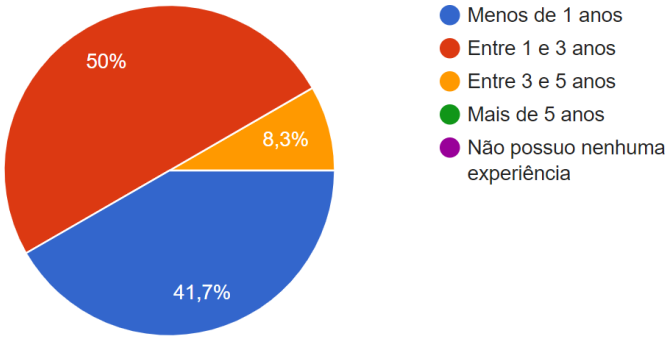
Figura 41 – Tempo de experiência em IoT dos participantes



Fonte: Elaborado pelo autor.

Referente à experiência dos participantes em computação em nuvem, 5 participantes possuem menos de 1 ano de experiência, 6 participantes possuem entre 1 e 3 anos e 1 participante possui entre 3 e 5 anos. Abaixo na Figura 42 é ilustrado os dados referentes aos participantes no que diz respeito a experiência em IoT.

Figura 42 – Tempo de experiência em computação em nuvem dos participantes



Fonte: Elaborado pelo autor.

Após fazer o levantamento dos perfis dos participantes é possível destacar o grau de conhecimento dos participantes em relação ao desenvolvimento de aplicações para CoT. É possível identificar que o perfil dos participantes se refere ao perfil de profissionais de desenvolvimento que não estão familiarizados com aplicações de IoT, pois o nível de conhecimento é pouco ou nenhum. Este perfil é satisfatório para compor a avaliação proposta, que visa responder a segunda questão desta pesquisa.

5.2.5 Resultados Obtidos

Nesta seção são apresentados os resultados obtidos na abordagem, que através da aplicação do questionário TAM foram coletadas informações sobre percepção de facilidade de uso, percepção de utilidade e intenção de comportamento, referente ao uso da abordagem MoT.

A Tabela 9 apresenta informações sobre os dados coletados referentes a facilidade de uso, percebida pelos participantes, onde 100% dos entrevistados concordam que o protótipo é fácil de usar e que não haveria necessidade de demasiado esforço para aprender as funcionalidades oferecidas pelo MoT (50% concordam parcialmente e 50% concordam totalmente). No caso de que não haveria dificuldades de se tornar um usuário perito, concordam que a abordagem é fácil de dominar (58,3% concordam totalmente e 41,7% concordam parcialmente).

Tabela 9 – Resultado referente a percepção da facilidade de uso

Afirmação	**Participantes**				
	Concordo Totalmente	**Concordo**	**Neutro**	**Discordo**	**Discordo Totalmente**
O protótipo do MoT fácil de usar	6	6	0	0	0
A abordagem do MoT é fácil de aprender	6	6	0	0	0
A abordagem do MoT é fácil de dominar	7	5	0	0	0

Com relação a percepção da utilidade, a Tabela 10 apresenta os dados coletados. Todos os participantes concordam que a abordagem facilitaria o desenvolvimento de aplicações de CoT

(58,3% concordam totalmente e 41,7% concordam parcialmente). Referente ao MoT contribuir para a produtividade dos desenvolvedores 58,3% dos participantes concordam totalmente e outros 41,7% concordam parcialmente. Com relação ao MoT proporcionar uma redução no tempo de entrega das aplicações de CoT 50% concordam parcialmente e 50% concordam totalmente. Quanto a abstração dos componentes de IoT reduzirem a complexidade do desenvolvimento de aplicações CoT, 50% dos participantes concordam parcialmente e 50% concordam totalmente. Da mesma, 58,3% concordam parcialmente que a abstração dos componentes de computação em nuvem reduz a complexidade das configurações dos ambientes, enquanto 41,7% concordam totalmente.

Tabela 10 – Resultado referente a percepção de utilidade percebida

Afirmação	**Participantes**				
	Concordo Totalmente	**Concordo**	**Neutro**	**Discordo**	**Discordo Totalmente**
A abordagem MoT facilitaria o desenvolvimento de aplicações de CoT	7	5	0	0	0
A abordagem MoT ajudaria na produtividade dos desenvolvedores	7	5	0	0	0
A abordagem MoT reduziria o tempo para entregar uma aplicação de CoT	6	6	0	0	0
A abstração dos componentes de IoT reduz a complexidade do desenvolvimento	6	6	0	0	0
A abstração dos componentes de Computação em Nuvem reduz a complexidade da configuração dos ambientes	7	5	0	0	0

A Tabela 11 apresenta os dados coletados referentes a percepção de uso geral, onde 100% dos participantes concordam que a abordagem poderia ser utilizada para desenvolvimento de aplicações de CoT na indústria de *software* (83,3% concordam parcialmente e 16,7% concordam totalmente). Quanto a capacidade de um usuário sem conhecimentos técnicos de IoT e computação em nuvem conseguir criar uma aplicação de CoT com o uso da abordagem MoT, 50% dos participantes concordam totalmente, outros 33,3% concordam parcialmente e apenas 16,7% ficaram neutros quanto essa afirmação. Da mesma forma, 83,3% dos participantes concordam que teriam intenção de usar a abordagem MoT para a construir uma aplicação de CoT, enquanto apenas 16,7% se mantiveram neutros nessa questão.

Tabela 11 – Resultado referente a percepção geral

Afirmação	**Participantes**				
	Concordo Totalmente	**Concordo**	**Neutro**	**Discordo**	**Discordo Totalmente**
A abordagem MoT poderia ser usada para desenvolvimento de aplicações de CoT na indústria de software	3	9	0	0	0
Uma pessoa com pouco conhecimento técnico em desenvolvimento de IoT e computação em nuvem poderia construir uma aplicação de CoT doméstica	5	6	1	0	0
Eu usaria a abordagem do MoT para construir uma aplicação de CoT	1	9	2	0	0

Portanto, os dados coletados e analisados sugerem que a abordagem MoT tem potencial de

aceitação por pessoas com perfis adequados com os dos participantes. Os resultados encorajam e mostram o potencial de usabilidade da abordagem proposta em um ambiente real.

5.3 Considerações finais

Nesta seção serão discutidos os resultados referente as duas avaliações da abordagem MoT apresentadas neste capítulo.

Referente a capacidade da abordagem de construir uma aplicação CoT, avaliada através do estudo de caso apresentado na Seção 5.1, pode-se concluir que abordagem não atingiu o objetivo. A execução da aplicação no ambiente de nuvem não foi possível devido a incompatibilidade entre os componentes da plataforma Node-RED com o ambiente de nuvem, o que impossibilitou a comunicação entre o sensor e a aplicação. No entanto, outras aplicações que usam outros componentes que foram criados diretamente pelo Node-RED não encontraram problemas para a execução.

Com base nos resultados obtidos a partir do estudo de caso, a primeira questão de pesquisa (QP-1) não foi respondida de forma satisfatória, pois a abordagem não atingiu a execução da aplicação no ambiente de nuvem. No mais a abordagem conseguiu criar os ambientes de nuvem para a execução, os serviços necessários para aplicação e também a geração do código-fonte para a plataforma Node-RED. Sendo assim, a abordagem ainda se mostra eficiente para a criação de aplicações CoT, e novas pesquisas podem estender o estudo para alcançar um ambiente de nuvem compatível com a aplicação criada, ou mesmo, implementar novas regras no transformador para alcançar outras plataformas de execução.

Referente percepção de usabilidade e aceitação dos usuários quanto ao uso da abordagem para a criação de aplicações CoT, 100% dos participantes concordam parcialmente ou totalmente que a abordagem facilitaria o desenvolvimento de aplicações, que a abordagem é fácil de dominar, e reduz a complexidade do desenvolvimento de CoT. Esses resultados apontam que a abordagem teve um boa aceitação pelos participantes do estudo experimental.

Finalizando pode-se concluir que a abordagem MoT e o protótipo apresentado, demonstraram que, com a execução da avaliação e os resultados obtidos através do questionário TAM, facilitam o desenvolvimento de aplicações para CoT mesmo para usuários com pouca ou nenhuma experiência em IoT e computação em nuvem, através da abstração dos detalhes de implementação propostos pela abordagem dirigida por modelos. Dessa forma respondendo de forma satisfatória à segunda questão de pesquisa (QP-2) elaborada neste estudo.

6 CONCLUSÃO

Este trabalho realizou uma análise das abordagens dirigidas por modelos usadas para o desenvolvimento de aplicações para IoT e que se beneficiam dos recursos da computação em nuvem. A partir da pesquisa realizada, é possível observar que já existe um relevante esforço no sentido de viabilizar o desenvolvimento de aplicações IoT através dessas abordagens. Com base nessa pesquisa pode-se concluir que as abordagens dirigidas por modelos oferecem soluções para lidar com as dificuldades existentes no processo de desenvolvimento de CoT.

Um abordagem de desenvolvimento dirigida por modelos denominada MoT, foi proposta para preencher as lacunas identificadas no contexto de aplicações CoT, principalmente no que diz respeita a heterogeneidade dos provedores de serviços de nuvem. Para por em prática tal abordagem, foi implementada uma ferramenta para dar suporte a essa abordagem. Cujo objetivo era responder as questões de pesquisa propostas apresentadas neste trabalho. A primeira buscou demonstrar se através de uma abordagem dirigida por modelos era possível construir aplicações de IoT nativas de nuvem com pouco ou nenhum esforço manual, e a segunda diz respeito a capacidade de pessoas com pouca ou nenhuma experiência de IoT conseguirem criar suas próprias aplicações através da abordagem proposta.

Um estudo de caso foi aplicado para validar a abordagem proposta, buscando responder a primeira questão de pesquisa. Ao qual apresentou resultados satisfatórios quanto a capacidade de criar uma aplicação de CoT, no entanto não conseguiu atingir a execução da aplicação no ambiente de nuvem. Uma segunda avaliação foi realizada seguindo o modelo TAM, para obter a percepção dos usuários quanto a utilidade e facilidade da abordagem, essa avaliação mostrou resultados que a abordagem tem um potencial de aceitação, mesmo por pessoas que tem pouca ou nenhuma experiência em IoT, respondendo positivamente a segunda questão de pesquisa proposta nesta pesquisa.

Por fim, pode-se concluir que a abordagem MoT tem um potencial para melhorar o desenvolvimento de aplicações de CoT, preenchendo algumas da lacunas identificadas na literatura. Os resultados obtidos na avaliação mostram que a abordagem teve uma boa aceitação por parte dos profissionais da área de tecnologia, mas que trabalhos futuros poderiam agregar resultados ainda melhores, tanto para o tema quanto para a pesquisa.

6.1 Contribuições

A realização dessa pesquisa trouxe quatro grandes contribuições referente ao tema de Desenvolvimento Dirigido por Modelos para aplicações de *Internet* das Coisas nativas de nuvem.

- **Abordagem dirigida por modelos:** A abordagem proposta nesta pesquisa buscou preencher algumas lacunas identificadas durante a análise do estado da arte. A abordagem MoT buscou atender todo o processo de desenvolvimento de uma aplicação de CoT, através de uma abordagem MDD. A complexidade dos elementos de IoT e as configurações

de serviços de computação em nuvem foram abstraídos para níveis mais elevados. Essa abordagem pode servir de referências para o desenvolvimento de novos trabalhos na área.

- **Análise do estado da arte:** Essa pesquisa realizou uma análise sobre o estado da arte, sobre desenvolvimento de aplicações para *Internet* das Coisas, e como abordagens MDD vem sendo aplicadas neste contexto de aplicações. Essa análise foi importante para identificar limitações, onde identificou-se oportunidades de pesquisas referentes a esse tipo de abordagem.

- **Ferramenta de transformação:** Este trabalho produziu uma ferramenta para suporte à abordagem proposta, capaz de auxiliar as equipes de desenvolvedores no processo de desenvolvimento de aplicações CoT, reduzindo o esforço dos envolvidos, de tal modo que venha tornar o desenvolvimento dessas aplicações mais ágil e fácil para que as empresas possam acompanhar o avanço das tecnologias e aumento da demanda de aplicações. Essa ferramenta é disponibilizado sob licença *open source* e poderá ser evoluída pela comunidade que tenha interesse em dar sequência ao estudo realizado neste trabalho.

- **Conhecimento empírico:** Essa pesquisa produziu conhecimento empírico sobre o desenvolvimento dirigido por modelos para aplicações de CoT. As avaliações realizadas neste trabalho, como o estudo de caso, que mostra o potencial que a abordagem pode oferecer quanto a automatização de etapas do desenvolvimento. E a avaliação seguindo o modelo de aceitação de tecnologia, que evidenciou a percepção de usabilidade e facilidade que esse tipo de abordagem pode oferecer aos desenvolvedores para aplicações CoT.

6.2 Limitações e Trabalhos Futuros

Este trabalho abordou algumas questões envolvendo uma abordagem dirigida por modelos para o desenvolvimento de aplicações CoT. As contribuições deste trabalho foram salientadas durante o trabalho apresentado, porém, é necessário mais pesquisas que possam fomentar este assunto e comprovar os resultados que foram encontrados nessa pesquisa. Há diversas possibilidades de criação de novos cenários para esta pesquisa, porém não há tempo hábil para que sejam adicionadas neste trabalho todas estas possibilidades. Com isto, é apresentado abaixo as limitações desta pesquisa e alguns pontos que são sugestões para trabalhos futuros:

- **Incluir novos diagramas UML:** A abordagem MoT foi desenvolvida com o objetivo de interpretar apenas diagramas de casos de uso dentre todos os diagramas presentes na UML. A implementação de outros modelos definidos pela UML, como o diagrama de classes, que possibilita a representação dos atributos, relacionamentos dos componentes, e o diagrama de atividades que poderiam definir funcionalidades mais complexas que envolveriam diversos estereótipos para representar um fluxo detalhado com alto nível de abstração.

- **Comparação entre a abordagem MoT e uma abordagem tradicional:** É possível a realização de uma comparação entre a abordagem MDD proposta pelo MoT com outras abordagens de desenvolvimento tradicionais utilizadas para IoT, a fim de obter novas métricas para comprovação da eficiência da abordagem MoT em relação as demais.

- **Aumentar o número de participantes:** A aplicação da avaliação deste trabalho foi realizada com público variado desde estudantes de graduação e pós-graduação dos cursos de tecnologia da informação a profissionais da indústria de *software*. Pode-se aumentar e estender a avaliação para o público em geral, a fim de avaliar a abordagem com participantes que não estão envolvidos com a tecnologia para verificar a aceitação da abordagem por pessoas comuns.

- **Avaliar a abordagem com profissionais de IoT:** Dos participantes envolvidos na avaliação da abordagem, existe pouca experiência de desenvolvimento de IoT, o que representa uma visão apenas de um público que possui pouca vivência com o desenvolvimento de IoT, então sugere-se a necessidade de uma avaliação da abordagem com um público de profissionais experientes nesse contexto de aplicações, para obter novas métricas para avaliação.

REFERÊNCIAS

AAZAM, M. et al. Cloud of things: integration of iot with cloud computing. In: **Robots and sensor clouds**. [S.l.]: Springer, 2016. p. 77–94.

AL-FUQAHA, A. et al. Internet of things: a survey on enabling technologies, protocols, and applications. **IEEE communications surveys & tutorials**, [S.l.], v. 17, n. 4, p. 2347–2376, 2015.

ANNUNZIATA, M.; EVANS, P. C. Industrial internet: pushing the boundaries of minds and machines. **General Electric**, [S.l.], 2012.

ARMBRUST, M. et al. **Above the clouds**: a berkeley view of cloud computing. [S.l.]: Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: a survey. **Computer networks**, [S.l.], v. 54, n. 15, p. 2787–2805, 2010.

CAI, H. et al. Model-driven development patterns for mobile services in cloud of things. **IEEE Transactions on Cloud Computing**, [S.l.], v. 6, n. 3, p. 771–784, 2018.

CANTANHEDE, R. F.; SILVA, C. E. da. Uma proposta de sistema de iot para monitoramento de ambiente hospitalar. **Anais da VII Escola de Computação e suas Aplicações-Epoca**, [S.l.], 2014.

CAVALCANTE, E. et al. On the interplay of internet of things and cloud computing: a systematic mapping study. **Computer Communications**, [S.l.], v. 89, p. 17–33, 2016.

CUBO, J.; NIETO, A.; PIMENTEL, E. A cloud-based internet of things platform for ambient assisted living. **Sensors**, [S.l.], v. 14, n. 8, p. 14070–14105, 2014.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, [S.l.], p. 319–340, 1989.

DELICATO, F. C.; PIRES, P. F.; BATISTA, T. **Middleware solutions for the internet of things**. [S.l.]: Springer, 2013.

FARIAS, C. M. de et al. Comfit: a development environment for the internet of things. **Future Generation Computer Systems**, [S.l.], v. 75, p. 128–144, 2017.

FOTEINOS, V. et al. Cognitive management for the internet of things: a framework for enabling autonomous applications. **IEEE vehicular technology magazine**, [S.l.], v. 8, n. 4, p. 90–99, 2013.

Gartner. **Gartner Says Organizations Must Update Their Network Access Policy to Address Attack of IoT Devices**, [S.l.], Sep 2016.

GIUSTO, D. et al. **The internet of things**: 20th tyrrhenian workshop on digital communications. [S.l.]: Springer Science & Business Media, 2010.

GUBBI, J. et al. Internet of things (iot): a vision, architectural elements, and future directions. **Future generation computer systems**, [S.l.], v. 29, n. 7, p. 1645–1660, 2013.

HERFS, W. et al. Model-based assembly control concept. In: EMERGING TECHNOLOGIES & FACTORY AUTOMATION (ETFA), 2013 IEEE 18TH CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p. 1–8.

JAISWAL, D. et al. A smart framework for iot analytic workflow development. In: ACM CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 13., 2015. **Proceedings...** [S.l.: s.n.], 2015. p. 455–456.

JOSEP, A. D. et al. A view of cloud computing. **Communications of the ACM**, [S.l.], v. 53, n. 4, 2010.

KLEPPE, A. G. et al. **The model driven architecture**: practice and promise. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 2003.

LAMPKIN, V. et al. **Building smarter planet solutions with mqtt and ibm websphere mq telemetry**. [S.l.]: IBM Redbooks, 2012.

LIKERT, R. A technique for the measurement of attitudes. **Archives of psychology**, [S.l.], 1932.

LUCRÉDIO, D. Uma abordagem orientada a modelos para reutilização de software. **INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO UNIVERSIDADE DE SÃO PAULO**, [S.l.], p. 37, 2009.

MARANGUNIĆ, N.; GRANIĆ, A. Technology acceptance model: a literature review from 1986 to 2013. **Universal Access in the Information Society**, [S.l.], v. 14, n. 1, p. 81–95, 2015.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. [S.l.]: 5. ed.-São Paulo: Atlas, 2003.

MESNIL, J. **Mobile and web messaging**: messaging protocols for web and mobile devices. [S.l.]: "O'Reilly Media, Inc.", 2014.

MEZGHANI, E.; EXPOSITO, E.; DRIRA, K. A model-driven methodology for the design of autonomic and cognitive iot-based systems: application to healthcare. **IEEE Transactions on Emerging Topics in Computational Intelligence**, [S.l.], v. 1, n. 3, p. 224–234, 2017.

NGUYEN, X. T. et al. Frasad: a framework for model-driven iot application development. In: INTERNET OF THINGS (WF-IOT), 2015 IEEE 2ND WORLD FORUM ON, 2015. **Anais...** [S.l.: s.n.], 2015. p. 387–392.

PATEL, P.; CASSOU, D. Enabling high-level application development for the internet of things. **Journal of Systems and Software**, [S.l.], v. 103, p. 62–84, 2015.

PEREIRA, C. R. **Aplicações web real-time com node. js**. [S.l.]: Editora Casa do Código, 2014.

PRAMUDIANTO, F. et al. Connecting the internet of things rapidly through a model driven approach. In: INTERNET OF THINGS (WF-IOT), 2016 IEEE 3RD WORLD FORUM ON, 2016. **Anais...** [S.l.: s.n.], 2016. p. 135–140.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico**: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª edição. [S.l.]: Editora Feevale, 2013.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G. **Unified modeling language reference manual, the**. [S.l.]: Pearson Higher Education, 2004.

SELIC, B. The pragmatics of model-driven development. **IEEE software**, [S.l.], v. 20, n. 5, p. 19–25, 2003.

SOSA-REYNA, C. M.; TELLO-LEAL, E.; LARA-ALABAZARES, D. An approach based on model-driven development for iot applications. In: IEEE INTERNATIONAL CONGRESS ON INTERNET OF THINGS (ICIOT), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p. 134–139.

THOMAS, D. Mda: revenge of the modelers or uml utopia? **IEEE software**, [S.l.], v. 21, n. 3, p. 15–17, 2004.

THRAMBOULIDIS, K.; CHRISTOULAKIS, F. Uml4iot—a uml-based approach to exploit iot in cyber-physical manufacturing systems. **Computers in Industry**, [S.l.], v. 82, p. 259–272, 2016.

VERAS, M. **Cloud computing**: nova arquitetura da ti. [S.l.]: Brasport, 2012.

VÖLTER, M. et al. **Model-driven software development**: technology, engineering, management. [S.l.]: John Wiley & Sons, 2013.

XIA, F. et al. Internet of things. **International Journal of Communication Systems**, [S.l.], v. 25, n. 9, p. 1101, 2012.

ZHOU, J. et al. Cloudthings: a common architecture for integrating the internet of things with cloud computing. In: COMPUTER SUPPORTED COOPERATIVE WORK IN DESIGN (CSCWD), 2013 IEEE 17TH INTERNATIONAL CONFERENCE ON, 2013. **Anais...** [S.l.: s.n.], 2013. p. 651–657.