UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia de Sorocaba
Campus de São João da boa vista

Joel Ravelli Junior

Arquitetura de Dados baseada no RAMI 4.0 para Manufatura Flexível na Indústria 4.0

Sorocaba
2022

Joel Ravelli Junior

Arquitetura de Dados baseada no RAMI 4.0 para Manufatura Flexível na Indústria 4.0

Dissertação de defesa apresentada como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica, junto ao Programa de Pós-Graduação em Engenharia Elétrica, interunidades, entre o Instituto de Ciência e Tecnologia de Sorocaba e o Campus de São João da Boa Vista da Universidade Estadual Paulista "Júlio de Mesquita Filho".

Área de concentração: Automação
Orientador: Prof. Dr. Eduardo Paciência Godoy

Sorocaba
2022

R252a

Ravelli Junior, Joel
Arquitetura de Dados baseada no RAMI 4.0 para Manufatura Flexível na Indústria 4.0 / Joel Ravelli Junior. -- São João da Boa Vista, 2022
102 p. : tabs., fotos

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Engenharia, São João da Boa Vista
Orientador: Eduardo Paciencia Godoy

1. Automação industrial. 2. Internet das coisas. 3. Interface de programação de aplicativos (software de computador). 4. Automação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Engenharia, São João da Boa Vista. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

unesp

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Sorocaba

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Arquitetura de Dados baseada no RAMI 4.0 para Manufatura Flexível na Indústria 4.0

AUTOR: JOEL RAVELLI JUNIOR
ORIENTADOR: EDUARDO PACIÊNCIA GODOY

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica, área: Automação pela Comissão Examinadora:

Prof. Dr. EDUARDO PACIÊNCIA GODOY (Participaçao Presencial)
Departamento de Engenharia de Controle e Automacao / Instituto de Ciência e Tecnologia - UNESP - Câmpus de Sorocaba

Prof. Dr. LEOPOLDO ANDRÉ DUTRA LUSQUINO FILHO (Participaçao Presencial)
Departamento de Engenharia de Controle e Automação / Câmpus de Sorocaba

Prof. Dr. MARCOSIRIS AMORIM DE OLIVEIRA PESSOA (Participaçao Virtual)
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos / Escola Politécnica da USP

Sorocaba, 16 de novembro de 2022

“Aqueles que se sentem satisfeitos sentam-se e nada fazem. Os insatisfeitos são os únicos benfeitores do mundo.”

Walter S. Landor

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Eduardo Godoy pela condução do meu trabalho.

A minha família e amigos, que me deram todo o suporte necessário.

Ao governo brasileiro, por oportunidades de estudar em instituições de excelência.

RESUMO

A Indústria 4.0 e Internet das Coisas (IoT) habilitaram a conectividade necessária para a transferência de informações, análises e controle de dados, possibilitando o aumento da produtividade dos negócios por meio de rápidas mudanças no escopo de produção em um mercado cada vez mais volátil. A inovação da IoT e Indústria 4.0 são vistas integrando sistemas de manufatura, gerenciando regras de negócios e descentralizando recursos de computação, permitindo mudanças rápidas por meio do uso da internet com segurança e confiabilidade. Para que isso aconteça, o Modelo de Arquitetura de Referência para a Indústria 4.0 (RAMI 4.0) foi elaborado, sistematizando os elementos da Indústria 4.0 em um modelo de camadas tridimensional. Um dos grandes desafios para a adoção do RAMI 4.0 é o desenvolvimento de soluções que suportem as funcionalidades de cada camada e as interações necessárias entre os seus elementos. Este trabalho objetiva o desenvolvimento de uma arquitetura de dados focada na Camada da Informação do RAMI 4.0 para aplicações de manufatura flexível na Indústria 4.0. Essa arquitetura foca na definição de um modelo da informação composto pelo banco e modelo de dados para estruturar e organizar a informação. Este trabalho demonstra também o desenvolvimento de um orquestrador para interconectar a comunicação industrial OPC UA com o framework de microsserviços utilizado para a criação de aplicações no formato de serviços em nuvem. Essa arquitetura na Camada da Informação viabiliza uma interface para fornecer ao cliente a capacidade de inserir, visualizar e até mesmo modificar um pedido em uma fábrica com base em suas necessidades e prioridades, proporcionando maior versatilidade e interoperabilidade na comunicação com as demais camadas superiores do RAMI 4.0. A arquitetura de dados foi testada e validada através de experimentos realizados num cenário de manufatura flexível com diferentes tipos de produtos, rotas e pedidos de produção. Os resultados demonstram como a arquitetura desenvolvida fornece uma perspectiva de produção com um serviço para manufatura flexível na Indústria 4.0, conectando sistemas de manufatura industrial com um sistema de informação para interface de negócio.

Palavras-chave: Automação industrial, Internet das coisas, Interface de programação de aplicativos (software de computador), Automação.

ABSTRACT

Industry 4.0 and the Internet of Things (IoT) have provided the connectivity required for information transfer, data analysis and control, enabling increased business productivity through rapid changes in the scope of production in a market increasingly volatile. The innovation of IoT and Industry 4.0 are applied in integrating manufacturing systems, managing business rules, and decentralizing computing resources, enabling rapid changes using the internet safely and reliably. For this to happen, the Reference Architecture Model for Industry 4.0 (RAMI 4.0) was developed, systematizing the elements of Industry 4.0 in a three-dimensional layered model. One of the great challenges for the adoption of RAMI 4.0 is the development of solutions that support the functionalities of each layer and the necessary interactions between its elements. This work aims to develop a data architecture focused on the Information Layer of RAMI 4.0 for flexible manufacturing applications in Industry 4.0. This architecture focuses on defining an information model composed of the database and data model to structure and organize the information. This work also demonstrates the development of an orchestrator to interconnect the OPC UA industrial communication with the microservices framework used to create applications in the form of cloud services. The architecture in the Information Layer enables an interface to provide the customer with the ability to enter, view and even modify an order in a factory based on their needs and priorities, providing greater versatility and interoperability in communication with the other upper layers of RAMI. 4.0. The data architecture was tested and validated through experiments in a flexible manufacturing scenario with different types of products, routes, and production orders. The results demonstrate how the developed architecture provides a production perspective with a service for flexible manufacturing in Industry 4.0, connecting industrial manufacturing systems with an information system for the business interface.

Keywords: Industrial Automation, Internet of Things, Application Programming Interface (Software Computing), Automation.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

ASS	Asset Administration Shell
API	Application Programming Interface
BPMN	Business Process Model and Notation
CLP	Controlador lógico programável
COAP	Constrained Application Protocol
COM	Component Object Model
CPPS	Cyber-Physical Production System
CPS	Cyber Physical Systems
DCOM	Distributed Component Object Model
EPC	Electronic Product Code
ERP	Enterprise Resource Planning
FDI	Field Device Integration
HDA	Historical Data Access
HW	Hardware
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I/O	Input/Output
IEC	International Electrotechnical Commission
IoT	Internet of Things
IIoT	Industrial Internet of Things
IoS	Internet of Services
IP	Internet Protocol
KPI	Key Performance Indicator
OPC UA	Open Protocol Communications Unified Architecture

OPC AE	Open Protocol Communications Alarms & Events
OPC HDA	Open Protocol Communications Historical Data Access
OPC DA	Open Protocol Communications Data Acquisition
OMG	Object Management Group
PLC	Programmable Logic Controller
PN	Part Number
PrSE	Production System Engineering
RAD	Rapid Application Development
RAMI 4.0	Reference Architecture Model for Industry 4.0
REST	Representational State Transfer
RFID	Radio Frequency Identification
SCADA	Supervisory Control and Data Acquisition
SDK	Software Development Kit
SI	Sistema Internacional de Unidades
SQL	Structured Query Language
SOA	Service oriented Architecture
SW	Software
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
UA	Unified Architecture
URL	Uniform Resource Locator
WEB	World Wide Web

SUMÁRIO

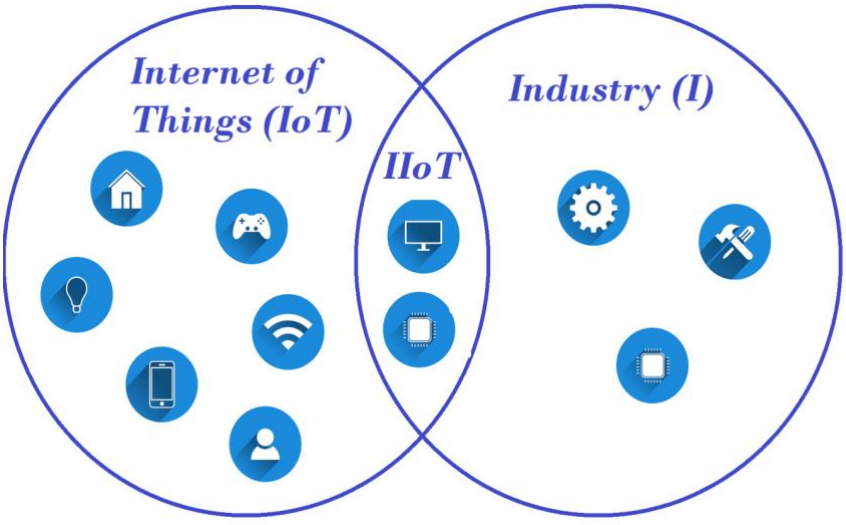
1- INTRODUÇÃO

1.1. Contextualização

Embasada nos avanços tecnológicos das últimas décadas, a tecnologia da informação veio moldando a sociedade como a conhecemos. Diversas tecnologias emergiram recentemente e mudaram a maneira como pensamos sobre o aumento da produtividade e redefiniram modelos de negócios. O surgimento da Internet das Coisas, do inglês *Internet of Things* (IoT), combinado com a relação entre inúmeros dispositivos, que se comunicam repetidamente para produzir e capturar dados que, ao serem consumidos e interpretados, geram *insights*, relatórios sucintos com informações chave para rápida interpretação e tomada de decisões (HASSANZADEH, 2015).

A evolução e o desenvolvimento das tecnologias digitais propiciaram a criação de novos métodos de produção nas indústrias globais baseados na automação, robótica, inteligência artificial, IoT e inteligência de dados, dentre outras inovações. A combinação entre a IoT e a indústria, representada pela Figura 1, derivou para a criação do conceito de Internet das Coisas Industrial (*Industrial Internet of Things - IIoT*).

Figura 1 - Elementos comuns entre IoT e industrial, permitem o surgimento de uma interseção entre a indústria tradicional e elementos de IoT, que viabilizam a IIoT.



Fonte: autoria própria.

IIoT foi definida pela utilização de tecnologias que afetam diretamente a automação em infraestrutura crítica e melhoram a produtividade empresarial (HASSANZADEH, 2015). A utilização dessas tecnologias no contexto industrial, coordenadas de modo a conferir competitividade ao negócio, otimizar a eficiência da cadeia produtiva, adicionar valor ao produto, racionalizar o uso dos recursos e customizar as soluções tecnológicas, é chamada de Indústria 4.0.

A IIoT tornou-se um dos elementos chave da Manufatura Inteligente e da Quarta Revolução Industrial, de acordo com Eruvankai (2017), a Indústria 4.0 foi introduzida pela primeira vez na feira de Hannover em 2011, devido a uma iniciativa do governo alemão adotada como parte do plano de ação estratégico de alta tecnologia para 2020 (KAGERMANN; WAHLSTER; HELBIG, 2013).

Os esforços citados pelo governo alemão e trabalhos da literatura se concentraram no estudo da estrutura do RAMI 4.0, para desenvolver soluções compatíveis com os requisitos de aplicações Indústria 4.0. Em Kagermann (2013) é apontado que essas demonstrações devem ser desenvolvidas e disponibilizadas ao mercado o mais rápido possível, considerando sua implementação estratégica e a adaptação das tecnologias e experiências básicas existentes aos requisitos de manufatura.

De acordo com Suri (2017), a Indústria 4.0 é uma das iniciativas mais proeminentes em direção à visão de manufatura inteligente para promover eficiência e sinergia entre fornecedores, produtores e clientes. A disponibilidade de uma ampla gama de dados e informações no contexto de Sistemas de Produção Cibernéticos-Físicos (do inglês *Cyber-Physical Production System CPPS*), oferecem uma grande possibilidade de criação de novas oportunidades de negócios. No entanto, para que essas oportunidades de negócios se materializem, com sucesso, é evidente a necessidade de uma comunicação eficiente e visualização clara de novas estratégias e do processo operacional subjacente entre todas as partes interessadas envolvidas na cadeia de valor da manufatura.

Promover a integração entre essas tecnologias é um dos grandes desafios da Indústria 4.0, que visa a evolução da realidade produtiva através da integração de dados e informações, além da tomada de decisões em tempo real. Muitos trabalhos têm focado no desenvolvimento de soluções que agreguem essas tecnologias e

permitam a integração dos processos produtivos e equipamentos de automação a serviços de TI que estão armazenados na nuvem.

Por esse motivo, o RAMI 4.0 surgiu propondo um modelo de arquitetura de referência para a Indústria 4.0, onde podemos destacar a sua arquitetura orientada a serviços ou SOA e eixos estruturais: níveis de hierarquia, ciclo de vida e fluxo de valor e camadas ou níveis. Esse diagrama tridimensional do RAMI 4.0 foi realizado pela Associação Alemã de Fabricantes Elétricos e Eletrônicos (*German Electrical and Electronic Manufacturers' Association - ZVEI*), para apoiar as iniciativas da Indústria 4.0, que estão alcançando ampla aceitação em todo o mundo. Conceitos, estrutura e métodos da Indústria 4.0 estão sendo adotados em todo o mundo para modernizar a manufatura (LYDON, 2019).

Na Figura 2, pode-se ver a interconexão entre objetos, serviços, pessoas e máquinas. A centralização de dados e informações viabilizou a quarta revolução indústria, através de uma maior perspectiva de controle em tempo real, viabilizando soluções automatizadas em larga escala.

Figura 2 - Integração entre processos para uma manufatura baseada em serviços e não em clientes.

Fonte: Autoria própria.

O grande desafio para adoção do RAMI 4.0 se encontra no desenvolvimento de soluções que suportem as funcionalidades de cada camada e as interações requeridas entre os elementos de cada camada. Nikolaidis et al. (2015) propõem a interação entre dois conceitos: a utilização de controladores em nuvem com dispositivos baseados em IIoT (NIKOLAIDIS, *et al.*, 2015). Enquanto, Wu et al. (2013) exploram o conceito de *Cloud Manufacturing*, que é um modelo de produção baseado em nuvem (WU D.; D., 2013), Chen et al. 2010 apresentam a ideia de *Robot as a Service* ou RAAS, sendo esse um serviço na nuvem para acesso a hardware e software de um robô (CHEN; DU; GARCÍA-ACOSTA, 2010).

Com base nas necessidades e desafios acima mencionados, este projeto se concentra nos estudos da Camada da Informação do RAMI 4.0 e interações de seus elementos para desenvolvimento de uma arquitetura de dados baseada no RAMI 4.0 para suportar aplicações de manufatura flexível na Indústria 4.0. A Camada da Informação foca no modelo da informação e estrutura dos dados, ou seja, como os dados são organizados e armazenados e como a informação é trafegada. O RAMI 4.0 permite a estruturação dos dados dentro da camada da informação, preferencialmente de maneira única, facilitando a disponibilização.

A manufatura flexível é um conceito que permite maior customização, fabricação de lotes menores sem perder eficiência, e produção mais ágil, através de uma produção assíncrona dentro do chão de fábrica. O produto não tem uma linha a percorrer para se tornar um bem final, ele navega pelas estações que contribuem para a construção do bem final, a partir de uma rota de produção específica para ele. Dessa forma, este trabalho foca na proposta de uma arquitetura de dados que simplifique entendimento na Camada da Informação.

Por isso, nesta proposta é utilizado o modelo de dados no formato JSON baseado no padrão proposto pelo *SensorThings API*, pertencente ao *Open Geospatial Consortium (OGC).* Esse modelo habilita criar interfaces de programação de aplicações, que utilizem o protocolo de comunicação HTTP, serviços RESTful e modelo de dados no formato JSON (BECKER, *et al.*, 2020). Além disso, é proposta a criação de uma conexão ponte (bridge) nesse trabalho, chamada de Orquestrador OPC UA, entre a comunicação industrial OPC UA e o framework Moleculer de microsserviços. Essa conexão criará uma interface permitindo ao cliente a capacidade de inserir, visualizar e até mesmo modificar um pedido com base em suas

necessidades e prioridades, permitindo que a indústria implemente mudanças rápidas para se adaptar ao mercado, originando um modelo de manufatura flexível, como um serviço em aplicações da Indústria 4.0.

Este trabalho faz parte de um projeto de pesquisa mais amplo do que é aqui apresentado. O foco macro é o desenvolvimento de uma arquitetura orientada a microsserviços baseada em todas as camadas (*Layers*) do RAMI 4.0, e este trabalho está especificamente focado na Camada de Informação, porém como possui grande sinergia com as camadas adjacentes (Comunicação e Funcional) do RAMI 4.0. Elas acabam sendo brevemente abordadas, integrando-se com todas as demais camadas do modelo, através de uma arquitetura baseada em serviços.

1.2. Objetivos

Este trabalho propõe o desenvolvimento de uma arquitetura de dados para a Camada da Informação do RAMI 4.0 para aplicações de manufatura flexível na Indústria 4.0. A integração de tecnologias é baseada no protocolo de comunicação OPC UA e no framework Moleculer de microsserviços para a criação de aplicações no formato de serviços em nuvem, proporcionando maior versatilidade e interoperabilidade em aplicações de produção como um serviço.

1.3. Estrutura e conteúdo

A estrutura deste trabalho ficou dividida em 6 capítulos, considerando esta introdução e desconsiderando as referências bibliográficas.

O Capítulo 2 trata de uma revisão da literatura que relaciona os principais tópicos do tema, como Indústria 4.0, estrutura da Camada da Informação para Indústria 4.0 e além de um estudo aprofundado de trabalhos pertinentes na área. A camada e modelagem da informação é um assunto de grande complexidade pois visa unificar aspectos fabris de maneira generalizada. Dentro desta contextualização, são apresentadas as pesquisas mais relevantes que foram publicadas com base no objeto de estudo proposto.

O Capítulo 3 apresenta os conceitos relacionados do RAMI4.0 e OPC UA, uma visão da arquitetura proposta para o desenvolvimento de uma interface controladora para Indústria 4.0 e a contextualização do RAMI4.0, explicando os

níveis de hierarquia e o fluxo de valor do ciclo de vida, juntamente com o OPC clássico e seguido pelo OPC UA.

O Capítulo 4 apresenta os materiais e métodos, uma visão da arquitetura proposta para o desenvolvimento de uma interface controladora para Indústria 4.0, juntamente com as tecnologias necessárias para a implementação da proposta.

O Capítulo 5 destaca os resultados experimentais obtidos com a implementação de software apresentado e as principais simulações, resultados e análises realizadas.

Por fim, o capítulo 6 apresenta as considerações finais e as conclusões retiradas da elaboração do trabalho.

2- REVISÃO DA LITERATURA

A seguir serão apresentados conceitos importantes para a compreensão do trabalho e discussões apresentadas.

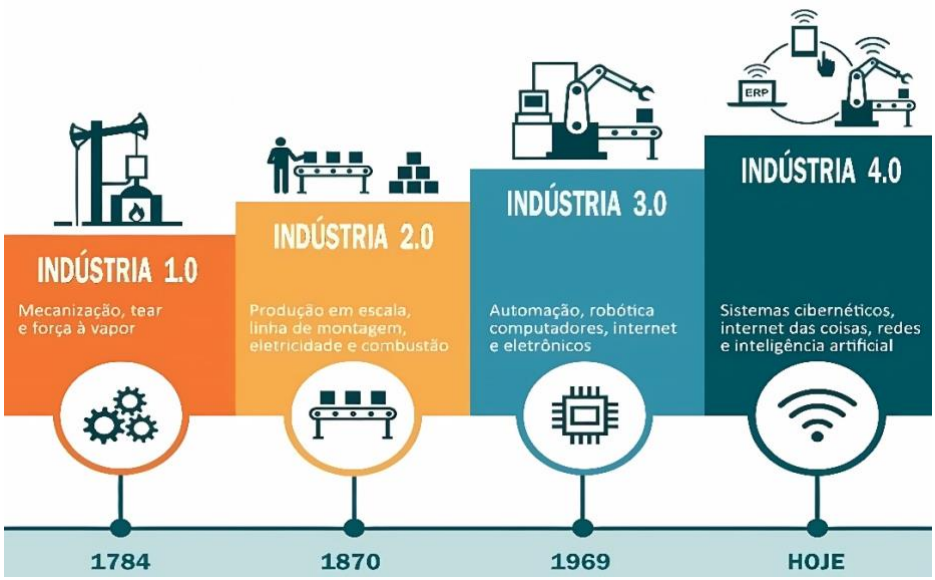
2.1. Indústria 4.0

A chamada quarta revolução industrial, também é conhecida como Indústria 4.0, na língua inglesa temos *Industry 4.0*. O número 4.0, cria uma referência a quarta revolução industrial que está acontecendo. A primeira Revolução Industrial é considerada principalmente a máquina a vapor que usou a energia do vapor explorável abrindo a era da indústria. A Segunda Revolução Industrial é vista principalmente como o uso de eletricidade para criar a produção em massa, principalmente na nova indústria automotiva (RÜTTIMANN, 2016). A Terceira Revolução Industrial está geralmente associada ao amplo uso da eletrônica e da tecnologia da informação para automatizar o processo de produção, também para possibilitar a manipulação de dados para manufatura integrada por computador (CIM – C*omputer Integrated Manufacturing*), trazendo para a era atual da tecnologia da informação e integração da tecnologia de operação (BASSI, 2017).

As revoluções industriais anteriores, sempre estiveram ligadas a invenções baseadas em descobertas científicas inovadoras, como Watt, Tesla, von Neuman, resultando na abertura de várias novas indústrias. Observe que, mesmo invenções realmente revolucionárias, como a telecomunicação sem fio de Marconi (Prêmio Nobel de 1909), que está na base da comunicação global atual, bem como todas as possibilidades por ela criadas e o moderno controle da cadeia de suprimentos da manufatura, não são consideradas revoluções para a indústria.

Portanto, o conceito da Indústria 4.0 não é uma revolução técnica ligada a uma descoberta revolucionária (RÜTTIMANN, 2016). A Indústria 4.0 organiza conexões especiais entre dispositivos físicos, informações, humanos e esferas biológicas para modificar estilos de vida tradicionais (BASSI, 2017). Na Figura 3 é mostrada uma linha do tempo com descrições e datas aproximadas de todas as revoluções industriais na história.

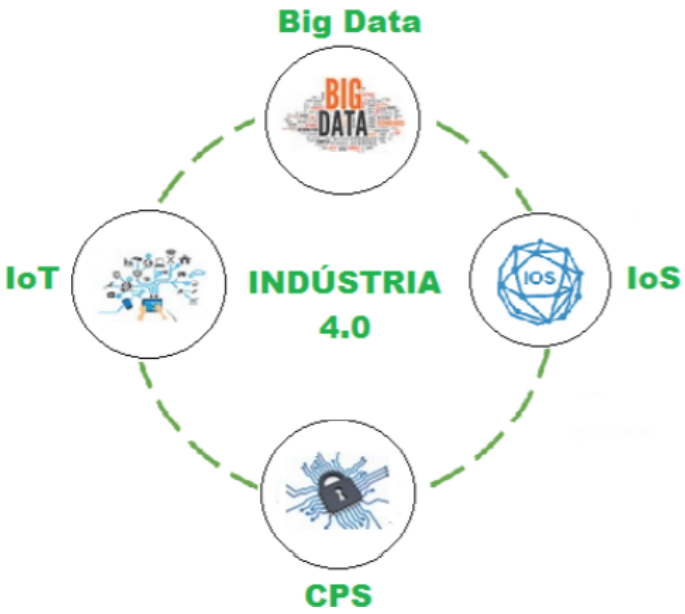
Figura 3 - Linha do tempo das revoluções industriais.



Fonte: (MELO, 2020).

Para viabilizar esse conjunto de características citadas anteriormente, a Indústria 4.0 integra-se às tecnologias da terceira revolução industrial, junto com componentes de processamento de informação que permitem gerenciá-las (MELO, 2020). Dentre as várias tecnologias que são contempladas para um treinamento 4.0, como realidade aumentada, linguagem em nuvem, framework, algumas se destacam como a Internet das Coisas (IoT), Big Data, Serviços da Internet e Sistemas Ciber-Físicos (TELLO; EDGAR, 2018), como mostra a Figura 4**.**

Figura 4 - As principais tecnologias que compõem a Indústria 4.0.



Fonte: Melo et al. (MELO, 2020).

O IIoT (*Industrial Internet Of Things*), nos últimos três anos continuou se movendo para cima e para baixo na ladeira do "gatilho de inovação" e, de acordo com o estudo Gartner (VELOSA, 2020), levará cerca de 5 a 10 anos para ganhar a adoção em massa, ou o chamado *mainstream* em um ambiente industrial, onde as linhas de produção não propriamente estão conectadas à internet, na maioria das vezes, e onde várias redes de empresas conectam independentemente, como por exemplo, o veículo automatizado guiado, serviços de manutenção, linhas de produção e automação de escritório (BASSI, 2017) e (MELO, 2020).

O Sistema Cyber-Físico (CPS), representa a integração de sistemas computacionais com processos físicos (LEE; SESHIA, 2017). Um CPS melhora de forma confiável, segura, eficiente e em tempo real, integrando comunicação e recursos de armazenamento de computador com monitoramento (SANISLAV; LIVIU, 2012). Um CPS é uma unidade de controle, geralmente um ou mais microcontroladores que controlam sensores e atuadores, sendo estes necessários para interagir com o mundo físico. Este conjunto de elementos também requer uma interface de comunicação para troca de dados com outros sistemas embarcados ou com a nuvem (JAZDI, 2014; MELO, 2020).

O termo Internet dos Serviços (IoS), refere-se à infraestrutura que fornece serviços aos consumidores (SCHROTH; JANNER, 2007). Consumidores e provedores podem negociar os serviços enquanto estão engajados em uma interação comercial, o que representa uma tecnologia de suporte para a visão de IoS (SCHROTH; JANNER, 2007). Diante dessa rede de clientes, eles podem ser atendidos por meio de um sistema onde as organizações trabalham em conjunto, sendo que valores são criados ao longo da cadeia de suprimentos da empresa, seus fornecedores, seus clientes e agregadores (MELO, 2020).

O Big Data pode processar grandes quantidades de informações coletadas diariamente no chão de fábrica (WITKOWSKI, 2017). O processamento e a análise dessas informações estão além da capacidade das ferramentas tradicionais utilizadas no cenário de TI, para dar conta dessa nova configuração, a tecnologia de Big Data é capaz de gerenciar e utilizar de forma rápida e eficiente a partir de um banco de dados em constante crescimento, ao mesmo tempo que permite a análise e separação do que é menos ou mais importante (MELO, 2020).

2.2. Estrutura da Informação para Indústria 4.0

Os sistemas de produção da Indústria 4.0 são conduzidos pela interpretação das rotas de produção, ou seja, a inteligência do controle não é mais codificada como era na fabricação tradicional. A engenharia é necessária na fase de definição do sistema de controle, adequando ao tradicional processo de design do sistema em cascata. Como os sistemas de produção da Indústria 4.0 são sistemas em cima de sistemas, sua estrutura e topologia levam a uma complexidade estrutural acentuada que é difícil de ser tratada (VYATKIN, 2013).

O RAMI 4.0, que será discutido nos capítulos a seguir, define uma arquitetura orientada a serviços onde os componentes fornecem serviços uns aos outros por meio de um protocolo de comunicação de rede. Os princípios básicos de SOA, que também serão aplicados em breve, são independentes de fornecedores, produtos e tecnologias. O objetivo é quebrar processos complexos em pacotes fáceis de entender, incluindo privacidade de dados e segurança de tecnologia da informação (TI) (LYDON, 2019).

De acordo com a ISA ou *International Society of Automation* (LYDON, 2019), as características do sistema de manufatura da Indústria 4.0 são:

- Sistemas e máquinas flexíveis;
- Funções distribuídas por toda a rede;
- Interação de participantes em níveis hierárquicos;
- Comunicação entre todos os participantes;
- Produto em fase de produção, como parte pertencente da rede;
- Estrutura RAMI 4.0.

Ainda de acordo com ISA, o influxo de tecnologia está começando a melhorar drasticamente a fabricação. No entanto, para fazer isso com eficácia é necessário planejamento, e o modelo RAMI 4.0 é um ponto focal para a compreensão de toda a cadeia de produção e fornecimento. Para entender apropriadamente os desafios futuros no campo, Soors, Ficzere e Dániel (SOOS, 2020) mostram que as direções de pesquisa atuais nas fábricas existentes, incluindo até mesmo as soluções de comunicação de baixo nível, devem ser reveladas. Do ponto de vista do provedor de serviços de comunicação, os recursos e parâmetros mais valiosos e críticos devem ser determinados e fixados no início da definição das dimensões da rede.

Os CPS são a base e permitem novos recursos em áreas como design de produto, prototipagem e desenvolvimento, controle remoto, serviços e diagnóstico, monitoramento de condições, manutenção proativa e preditiva, rastreamento, integridade estrutural e monitoramento de integridade de sistemas, planejamento, entre outros. A indústria 4.0 se baseia em modelos de dados e mapeamento de dados em todo o ciclo de vida e valor de produto de ponta a ponta mencionados. Todas as tecnologias na Indústria 4.0 precisam ser vistas nessa perspectiva em que a integração é a chave. Nos próximos parágrafos desta seção, serão apresentados trabalhos pertinentes a camada da informação do RAMI 4.0 com foco no estudo e entendimento de proposta para superação dos desafios apresentados.

Lydon (2019) nos mostra tentativas de preencher o espaço entre o domínio físico e digital, que incluem a implantação de soluções personalizadas para buscar, armazenar e analisar dados. As empresas começaram a centralizar a coleta de dados de sistemas de manufatura criando os *data warehouses* para ter uma única fonte de dados estruturada para análise de negócios. Como mostrado por Gandomi (2015), citado por Lydon (2019), na produção industrial, que é impulsionada pelo medo de perder conhecimentos ou intuições ocultas de dados de chão de fábrica, todas as fontes de dados estão sendo identificadas como potencialmente úteis e, portanto, são preventivamente registradas.

As tendências mostram que a indústria de manufatura está convergindo com objetivos da Indústria 4.0, implementando soluções para os diferentes aspectos de coleta, armazenamento, integração, descoberta e análise de dados, deixando incertezas sobre a implementação individual, bem como sua interoperação. Para superar esse problema, diferentes arquiteturas de referência da Indústria 4.0 foram propostas, fornecendo diretrizes sobre como implementar soluções gerais para fazer a ponte entre o domínio físico e digital.

Essas arquiteturas precisam apoiar a coleta de conjuntos de dados apropriados e fornecer feedback sobre os conjuntos de dados para o provedor de dados associado, a fim de se adaptar aos requisitos do caso de uso de análise (KIRMSE, 2019). No entanto, como o RAMI 4.0 é apenas uma estrutura abstrata, há uma necessidade de arquiteturas concretas que mostrem como implementar *data lakes* para o caso de uso de analítica de dados nas aplicações de 4.0.

Os autores apresentam uma abordagem para a implementação de uma arquitetura em conformidade com RAMI 4.0, lidando com a conexão de fontes de

dados industriais a um sistema de armazenamento de *data lake* e usando os dados para casos de uso analítico. Considerando o enriquecimento semântico, adicionando modelos semânticos durante a fase de integração dos agentes de ingestão e da entrada do usuário, o uso de dados considera a extração semântica pelos agentes correspondentes, proporcionando tarefas de transformação semântica e estrutural. O uso combinado dos agentes é utilizado em um ciclo analítico de dados estendido, fornecendo aplicativos com uma integração próxima aos processos de negócios.

Uma série de arquiteturas de alto nível foram propostas para atender aos desafios mencionados (KIRMSE, 2019). Uma das arquiteturas de alto nível é o chamado "*pipeline* de *big data*" publicado em Labirintis (2012), os autores descrevem um processo serial de várias fases, que são etapas necessárias para permitir a análise de dados. Os *pipelines* fornecem aos leitores uma visão geral básica sobre como extrair valor de *big data*, mas não descreve como vincular as várias fases no *pipeline*, nem como implementar os conteúdos das fases individuais.

Semelhante ao RAMI 4.0, o "*Industrial Data Space*" se concentra principalmente na descrição de diferentes funções dentro de um "ecossistema de dados". Cada função possui diferentes obrigações, dependendo da camada. A autorização de uso de dados é uma tarefa para o proprietário dos dados na camada funcional. Essa arquitetura de referência estabelece funções e atribui responsabilidades no espaço de dados em um nível superior, mas não lida com a localização.

Neste artigo, os autores discutem como os desafios de acessibilidade de dados, descoberta e interoperabilidade de conjunto de dados se manifestam ao implementar um conceito como o RAMI 4.0. Descrevendo com grande profundidade a realização do conceito com base em uma implementação usando uma abordagem baseada em agente. A abordagem é centrada em torno de um sistema de *data lake*, considerando a heterogeneidade das fontes e ativos de dados industriais, ao fornecer mecanismos para enriquecer semanticamente essas fontes, descrevendo seus conteúdos e formatos durante a fase de ingestão. Para a utilização dos dados, os autores descrevem como os dados podem ser acessados e processados semanticamente por agentes de extração auto-adaptáveis, fornecendo interfaces flexíveis para aplicativos orientados a dados.

O artigo de Lydon (2019) traz uma perspectiva profunda sobre *data warehouse* no contexto da indústria 4.0, considerando a conexão e usabilidade no contexto do

RAMI 4.0. Mesmo que o trabalho traga uma excelência na compreensão dos conceitos, faltou um exemplo implementado ou simulado para a proposta em questão.

O artigo de Novak (2019) traz uma visão da engenharia dos sistemas de produção da Indústria 4.0, que requer alta flexibilidade e curtos ciclos de atualização para sincronizar os resultados de grupos de trabalho colaborativos paralelizados. A troca de dados tradicional não é adequada, pois torna difícil manter uma visão geral das versões dos artefatos de engenharia. Para lidar com esses desafios de flexibilidade, complexidade e atualização rápida, a proposta aborda as seguintes questões de pesquisa:

1. Quais são as principais funções em um projeto de engenharia de sistema de produção típico para a Indústria 4.0, e quais informações são trocadas?

Para abordar a questão 1 os autores aplicaram o método de análise de processo de na unidade de teste dos autores em Praga como base para derivar um modelo de processo das principais funções e tarefas na engenharia de sistema de produção (*PrSE - Production System Engineering*) e a maioria dados relevantes que eles trocam.

2. Qual é um modelo de informação apropriado para troca de dados em um projeto de engenharia de sistema de produção típico da Indústria 4.0?

Para responder a segunda questão, os autores modelaram conceitos comuns que duas ou mais funções compartilham no processo de PrSE em um modelo de informação inicial, como base para orientar o projeto de um repositório de artefatos de engenharia que os especialistas de domínio podem enriquecer ao longo do processo de PrSE.

Os sistemas de produção contam com várias tecnologias e linguagens de programação, que devem funcionar juntas de maneira confiável por muitos anos. A lógica de controle central dos sistemas de produção é normalmente implementada em Computadores Logico Programável (CLP). Uma família de linguagens de programação é padronizada como IEC 61131. A robótica industrial utiliza linguagens proprietárias, como a linguagem KRL para robôs industriais tradicionais KUKA2 ou a linguagem Java estendida para robôs KUKA cooperativos. Níveis de automação mais altos utilizam linguagens de programação mais convencionais.

Modelagem e formalização de processos, de acordo com Allweyer (2016), como citado por Novak (2019). O Modelo de Processo de Negócios e Notação 2.0 (BPMN 2.0), busca fornecer uma abordagem de modelagem fácil de entender com

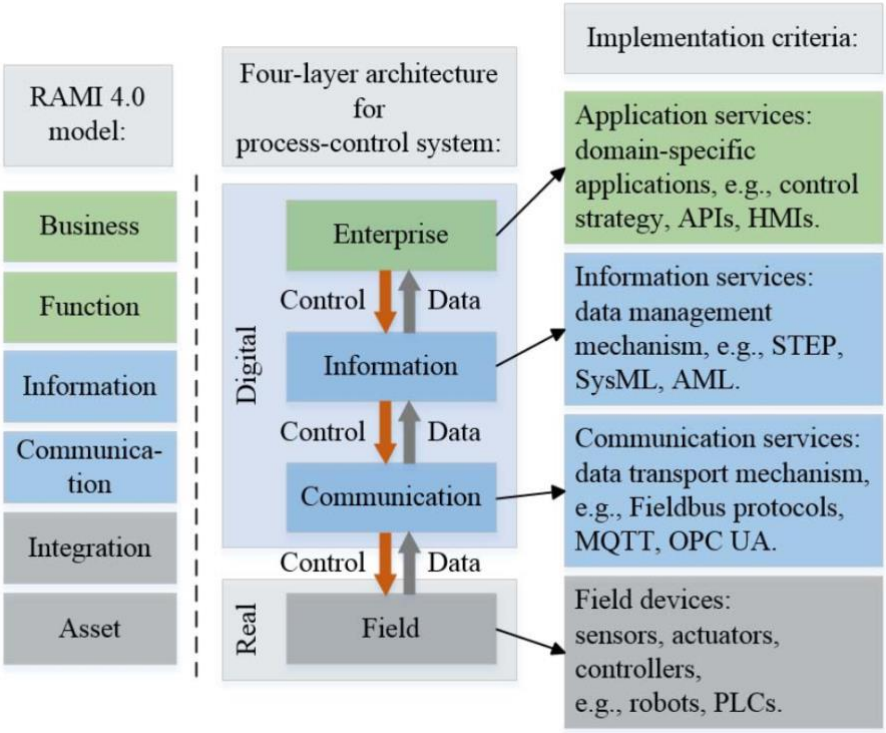
semântica rica. Para atingir esse objetivo, o BPMN fornece conceitos bem definidos, incluindo tarefas, eventos, gateways e comentários. As chamadas raias de natação permitem modelares as responsabilidades de diferentes atores ou grupos de trabalho, destacando as interfaces potenciais entre os especialistas do domínio. Conceitos semelhantes também podem ser modelados em diagramas de atividades UML.

Os autores observaram que os processos de engenharia tradicionais não se adequam aos sistemas de produção em conformidade com a Indústria 4.0. O artigo apresenta uma nova metodologia de projeto que reflete as necessidades no contexto da Indústria 4.0, relacionadas à alta flexibilidade, trabalho de ida e volta paralelo dos engenheiros e ciclos curtos de atualização. No contexto da Indústria 4.0, obter e manter uma visão geral sobre as mudanças e o status do projeto geral de engenharia é de grande importância. Para abordar a questão de pesquisa, os autores identificaram os principais papéis da engenharia e especificamos quais informações são. Com base nisso, foi desenhado o modelo de informação para a troca de dados.

A proposta apresentada é um dos poucos artigos focados em modelagem de dados para a indústria 4.0 e trouxe uma perspectiva interessante focando nos papeis dos profissionais da indústria para a elaboração do modelo. Porém deixa de lado aspectos deveras importantes como a padronização de dados e tecnologia, além da modelagem da informação, deixando um vácuo grande na proposta.

Analisando o artigo de Ye (2018), onde propõem uma solução baseada no uso do AutomationML e OPC UA para processos de manufaturas da indústria 4.0. Esse artigo foi analisado para nortear os estudos iniciais desse trabalho e propor alternativas, pois a solução do sistema é comumente vista na literatura, com algumas importantes alterações. A integração proposta é um ponto forte desse artigo, pois utiliza várias tecnologias de sistemas ciberfísicos, dos quais podemos destacar a implementação do um *shell* administrativo, utilizando *Automation Markup Language* ou AutomationML (AML) e OPC UA. Um sistema experimental também foi proposto para apoiar a ideia dos autores, assim desenvolver uma arquitetura refinada do modelo RAMI 4.0, que propõem quatro camadas e elas são: “*Enterprise layer*”, “*Information layer*”, “*Communication layer*” e “*Field layer*”. Estas camadas propostas possuem correspondência direta com as camadas do RAMI 4.0, conforme podemos observar na Figura 5.

Figura 5 - Arquitetura refinada de quatro camadas para sistemas de manufatura.



Fonte: Ye et al. (YE; HO HONG, 2018).

Enterprise Layer: a camada *Enterprise Layer* agrega duas camadas de *Business* e *Functional* do modelo RAMI 4.0 e diz respeito as regras de negócio, APIs e interfaces de usuário para acesso a todos os recursos das camadas inferiores.

Information Layer: a camada *Information Layer* tem a correspondência da camada do mesmo nome do modelo RAMI 4.0. Está camada armazena todos os dados e informações sobre o negócio, para tomada de decisões e incluí ainda a representação virtual estruturada de ativos, nos apresentando um exemplo de implementação da camada de informação para embasar esse trabalho.

Communication Layer: a camada *Communication Layer* é responsável pela entrega de dados ou informações entre os dispositivos e aplicações dos usuários. Para isso, é empregado o uso de softwares com comunicação máquina a máquina, da expressão em inglês *machine-to-machine* (M2M), como o OPC UA ou MQTT, que funcionam como um *broker* de mensagens, com a finalidade de receber e entregar mensagens obedecendo certas regras atribuídas, garantindo segurança e desempenho. Esta camada tem a mesma função da camada de *Communication* do RAMI 4.0.

Field Layer: a camada *Field Layer* incorpora as camadas *Integration* e *Asset* do modelo RAMI 4.0, que mapeia elementos físicos no sistema de manufaturas, como

sensores, CLPs, atuadores e entre outros. Estes elementos cooperam entre si para executar tarefas organizadas, concordando com a camada de *Enterprise* do modelo proposto.

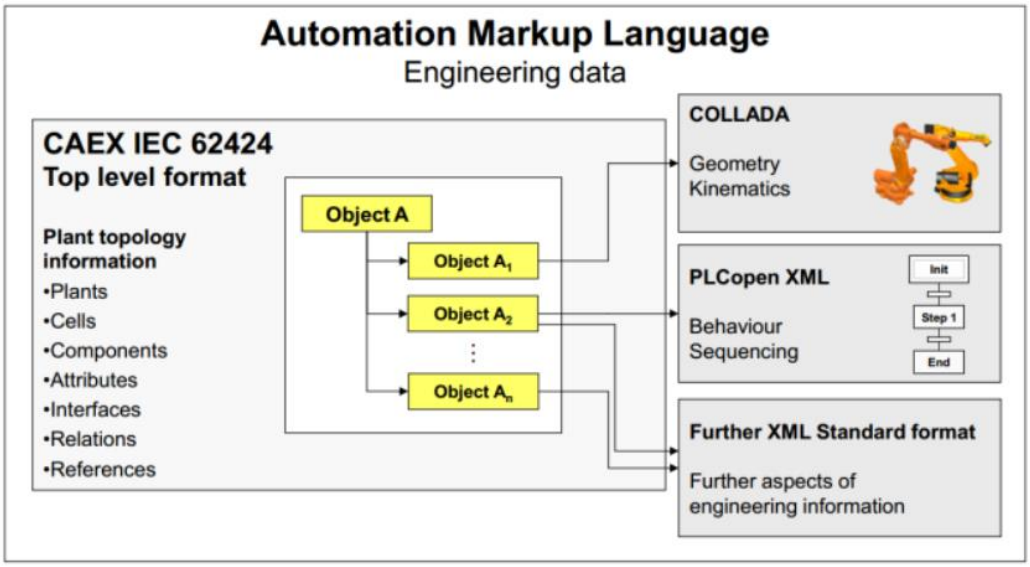
2.3. Padrões e Modelos de Dados

Nesse tópico é apresentado o estudo realizado de modelos da informação compatíveis para nortear a proposta escolhida de utilização do modelo de dados adaptado proposto pelo *OCG SensorThings API*, onde o formato JSON alinhado com as boas práticas de modelagem UML e a intercambialidade entre diferentes formatos como UML e XML, demostraram a viabilidade da proposta de trabalho para a estruturação da camada da informação. Então, os tópicos abaixo apresentam uma análises viáveis para a estruturação da camada da informação, passando desde o formato XML, apresentado na seção sobre o *AutomationML*, até a proposta do *OCG SensorThings API* e seu modelo de dados no formato JSON. As formas de transformações dos dados entre diversos formatos baseados em UML estão nas seções a seguir.

2.3.1. AutomationML

No ambiente fabril, há diversos tipos de sistemas especializados e muitas vezes com seus formatos de dados próprios. O AutomationML foi pensado para propor uma padronização do formato de dados consequentemente, padronizar as comunicações entre sistemas, que pode ser utilizado para produtos, processos e para descrição de recursos do sistema de produção do domínio da engenharia. A Figura 6 nos mostra o formato geral do AML que adapta, estende e combina outros padrões de formato de dados, em uma forma estruturada, incluindo os padrões CAEX, COLLADA, e PLCOpen XML.

Figura 6 - Visão geral da linguagem de marcação de automação.



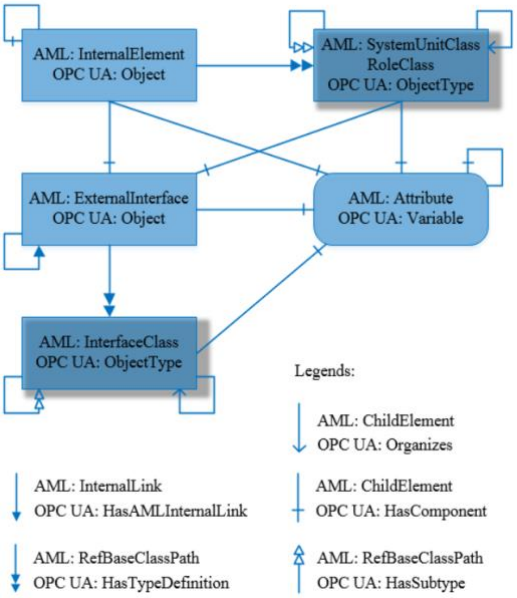
Fonte: Ye et al. (YE; HO HONG, 2018).

O formato de dados AML é consolidado tanto no meio acadêmico quanto na indústria e o seu formato de documento possui vantagens e desvantagens. Duas importantes desvantagens do AML, que utiliza o padrão XML, é ter uma estrutura de dados complexa e desempenho menor na análise e transformação dos dados computacionalmente falando. Então, outros formatos veem se popularizando, como o caso do JSON e YAML. (BECKER, *et al.*, 2020; ROBIN, 2018).

Nesse artigo, a implementação do sistema leva em consideração o OPC UA na camada de comunicação e AML para a camada de informação, com a comunicação de transferência de dados no formato binário.

Na Figura 7, temos um exemplo do mapeamento em alto nível das regras entre o AML e OPC UA, onde as estruturas padrões OPC UA correspondem a estrutura AML. Isso garante a entrega dos dados para a camada da informação e o armazenamento em bancos de dados temporais. Então, nesse esquema temos o AML *InternalElement* mapeado no objeto OPC UA e o AML *Attribute* torna-se uma variável no OPC UA e os tipos de classe AML (caixas quadradas), são transformados em OPC UA *ObjectType*. Diferentes tipos de setas representam relacionamentos mapeados entre objetos OPC UA, que implica na simplificação da criação do modelo de informações OPC UA, com base nos dados AML já previamente modelados e bem definidos.

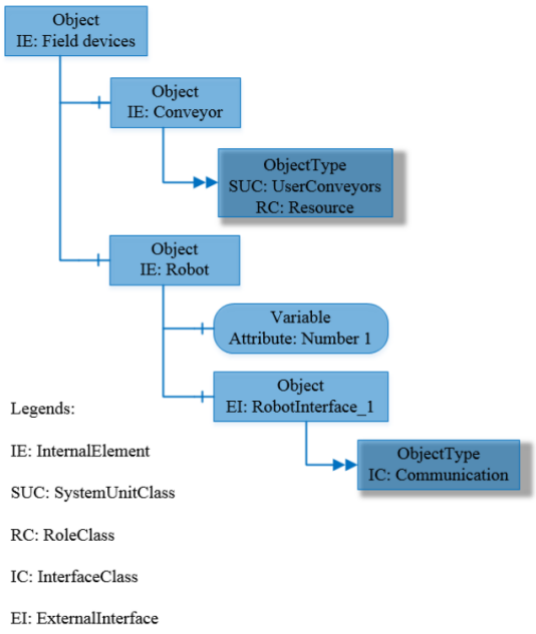
Figura 7 - Regras de mapeamento entre AML e o OPC UA.



Fonte: Ye et al. (YE; HO HONG, 2018).

Na Figura 8, temos um exemplo do mapeamento de dados do sistema em AML da solução. Podemos verificar o objeto dos dispositivos (*Field devices*), na camada *Field*, mapeado em AML com suporte a dois objetos físicos *Conveyor* e *Robot*. Cada um deles possuem seus próprios elementos, como atributos, regras de classe, interfaces e dependências. Pode-se afirmar que essa estrutura é complexa se comparado a outras estruturas, como o JSON (BECKER, *et al.*, 2020; ROBIN, 2018).

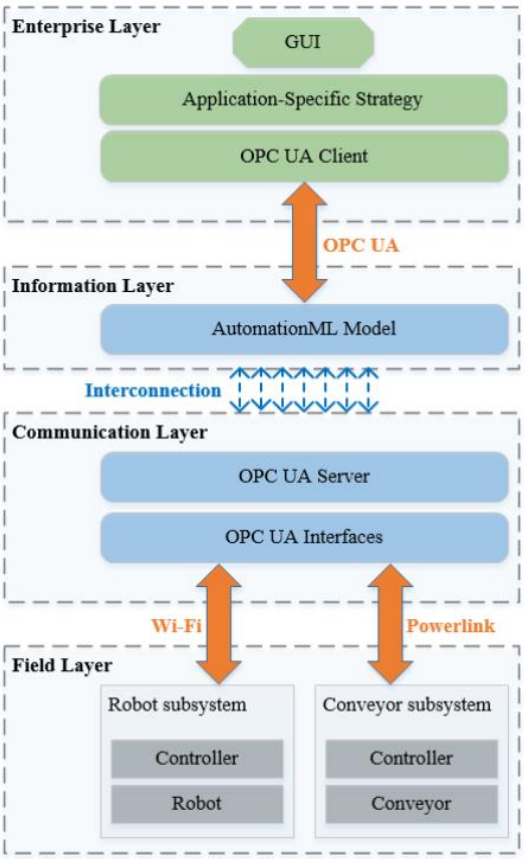
Figura 8 - Exemplo de AML mapeado e OPC UA.



Fonte: Ye et al. (YE; HO HONG, 2018).

O trabalho do artigo de Ye e Ho Hong (2018) continua com a implementação da arquitetura refinada proposta para o sistema da manufatura, baseado no modelo de referência RAMI 4.0. Pode-se observar na Figura 9 a sua arquitetura macro, onde o foco fica por conta do formato de dados e seu fluxo entre as camadas de Information e Communication, já discutidos acima nas Figura 7 e Figura 8. Também, pode-se observar o mapeamento dos objetos reais presentes na camada Field, Robot e Conveyor, que irão gerar os dados ou serem controlados pelas camadas superiores.

Figura 9 - Diagrama do modelo refinado para um sistema de manufaturas.



Fonte: Ye et al. (YE; HO HONG, 2018).

Segundo (YE; HO HONG, 2018), duas questões foram encontradas relativas à implementação do sistema. A primeira é como integrar os dados de um robô ou um *conveyor* utilizando AML, que foi solucionado com a estruturação correta dos parâmetros e características do objeto no arquivo AML, adição de identificadores únicos para cada objeto, o correto encapsulamento lógico ou físico de relações entre elementos e adição dos atributos de cada objeto.

Podemos observar que, as características de mapeamento entre o físico e digital podem ser aplicadas em outros formatos de dados, como o JSON e YAML. A segunda questão é sobre a utilização de diferentes tipos de redes para comunicação

entre a camada *Field* e *Communication*, conforme presente na Figura 9, onde podemos ver dois tipos diferentes de redes, que são a *Ethernet Powerlink* e Wifi. Esta última questão foi resolvida com a introdução de um *backbone* da rede industrial na fábrica, que oferece conectividade para ambos os tipos de rede, endereçamento de rede e o uso da arquitetura SOA, que é inerente ao servidor OPC UA.

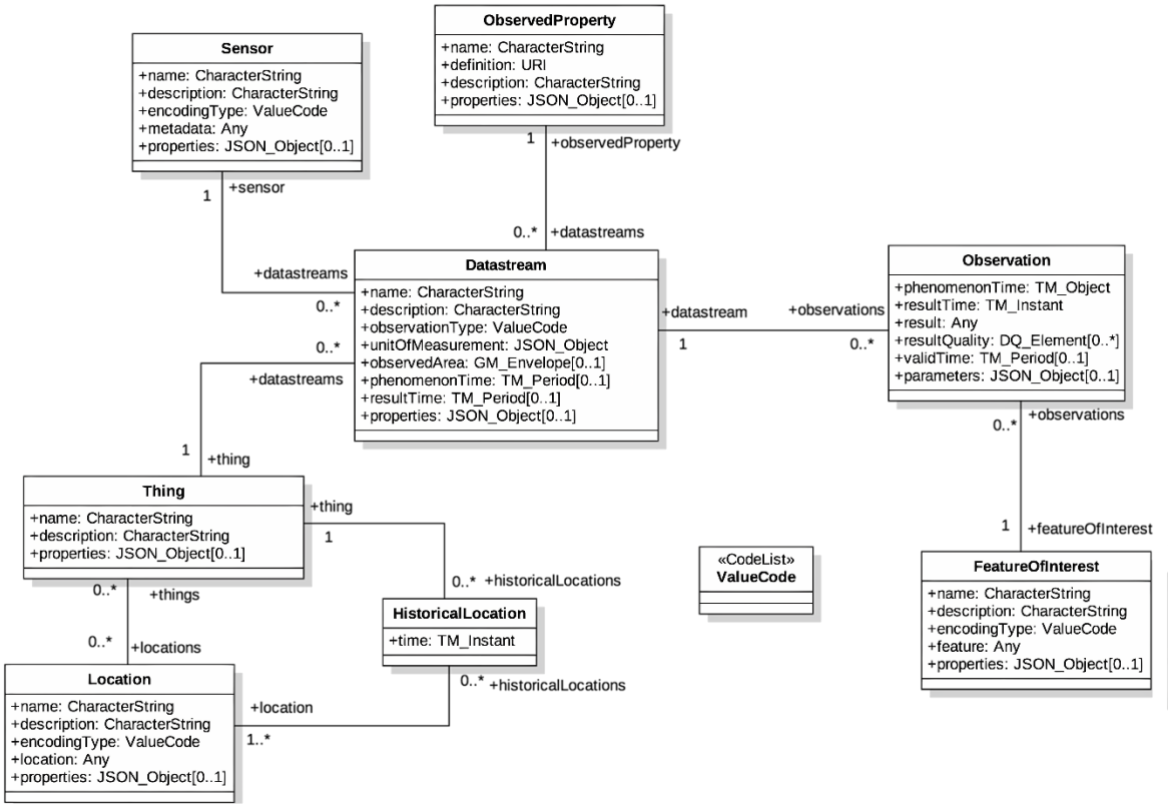
2.3.2. OCG SensorThings API

Em Becker (2020) é descrito os principais padrões para fábricas inteligentes com a plataforma *Smart Factory Web* (SFW), que possui o objetivo de conectar fábricas para permitir o uso flexível de ativos em uma cadeia de suprimentos. Os padrões SFW são aplicados em: comunicações entre fábricas e o SFW; comunicação entre fabricas e outros ambientes em nuvem para monitorar a produção; descrever as capacidades da fábrica e comunicação entre fabricas para troca de informações de produtos, realizar pedidos e compartilhar informações. Também, o SFW oferece um guia para implementações do OPC UA, *AutomationML* e OCG *SensorThings API*. Becker (2020) aborda dois tópicos essenciais a esse trabalho: as recomendações *OCG SensorThings API* e Assets na Indústria 4.0.

O padrão SensorThings API pertence ao *Open Geospatial Consortium* (OGC), e é pensado para criar interfaces de programação de aplicações, do inglês *Application Programming Interface (API)*, que utilizem o protocolo de comunicação HTTP, serviços RESTful e modelo de dados no formato JSON. No entanto, existem adaptações para o padrão *SensorThings* API funcionar com o protocolo de comunicação MQTT (LIANG; HUANG; KHALAFBEIGI, 2016).

No diagrama UML da Figura 10, temos as relações propostas do modelo de dados. A entidade *Thing* é o ponto central do modelo de dados que pode ser tanto um objeto físico ou virtual e possuir um ou mais sensores mapeados, por exemplo, um sensor de temperatura ou um carro. A entidade *Datastream*, responsável pelos dados coletados pertencentes a entidade *Thing*, possui uma coleção de leituras em *Observation*, que mede um valor presentes na entidade *ObservedProperty* e produzida por um Sensor associado.

Figura 10 - Diagrama UML OCG SensorThings API.



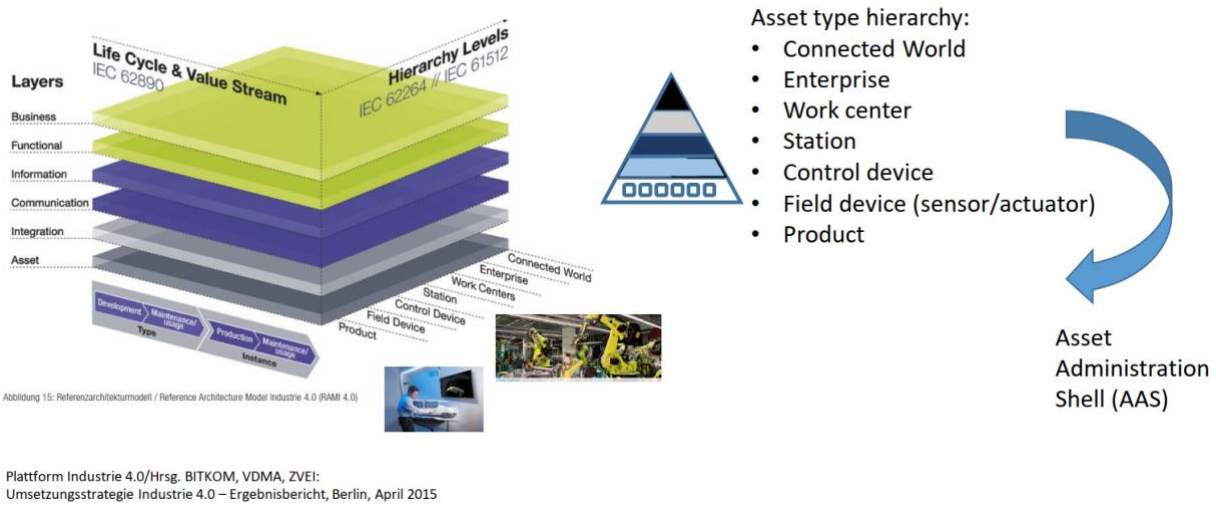
Fonte: Becker et al (BECKER, *et al.*, 2020).

2.4. Asset Administration Shell

Os níveis de hierarquia do RAMI 4.0 são: *Connected World*, *Enterprise*, *Work Centre*, *Station*, *Control Device*, *Field Device* e *Product*, de acordo com Reference Architecture Model Industrie 4.0 (Reference Architecture Model Industrie 4.0 (RAMI4.0), 2016) e a Figura 11, citado por (BECKER, *et al.*, 2020). Assim, o *Connected World* e *Product* vai além dos níveis clássicos da pirâmide de automação e representam a chave da Indústria 4.0. A combinação do *Asset* e a representação virtual com o *shell* administrativo, diz respeito a um componente da Indústria 4.0 (BECKER, *et al.*, 2020).

Pode-se destacar que o Asset representa um modelo virtual no *shell* administrativo e pode ter várias representações virtuais, ou seja, vários outros *shell* administrativos, serviços ou item físico. Cada Asset possui sua identidade e vida útil (BECKER, *et al.*, 2020).

Figura 11 - Modelo da Arquitetura de Referência Indústria 4.0 (RAMI 4.0).



Plattform Industrie 4.0/Hrsg. BITKOM, VDMA, ZVEI:
Umsetzungsstrategie Industrie 4.0 – Ergebnisbericht, Berlin, April 2015

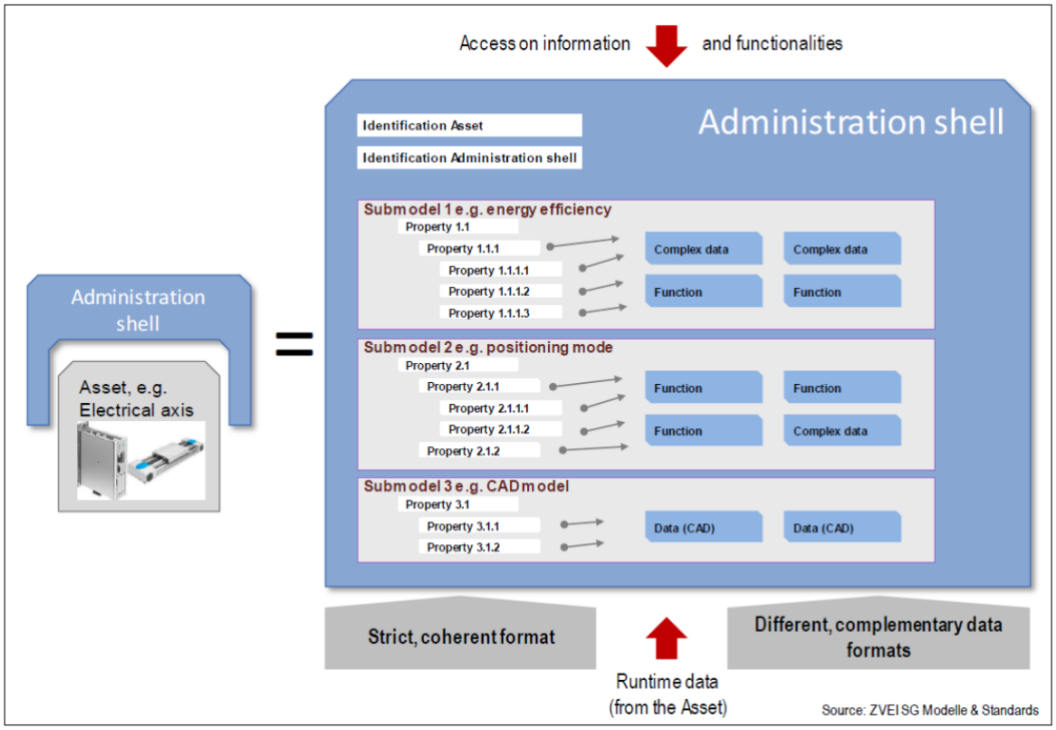
Fonte: Becker et al (BECKER, *et al.*, 2020).

2.4.1. Submodelos e Propriedades

O *Asset Administration Shell (AAS)* possui uma estrutura baseada em submodelos e compreende um conjunto de propriedades estruturadas, que segue uma hierarquia definida (BECKER, *et al.*, 2020). O submodelo possui suas propriedades que representam dados ou funções. O formato deve seguir as padronizações das normas IEC 61360-1 / ISO 13584-42 (IEC 61360-1, 2017; ISO 13584-42, 2010), para as propriedades e os valores as normas ISO 29002-10 e IEC 62832-2 (ISO/TS 29002-10, 2009; IEC 62832-2, 2020).

Na Figura 12, temos a estrutura básica de um AAS, que é representada pelo *Asset* em conjunto com o *shell*, e sua representação virtual. Suas propriedades, representada pela palavra em inglês *Property*, posicionados dentro das classes, chamadas de submodelo, do inglês *submodels*. O *shell* administrativo é alimentado pelos dados que vêm diretamente do *Asset*, descrito na seta vermelha da Figura 12, em *Runtime data*.

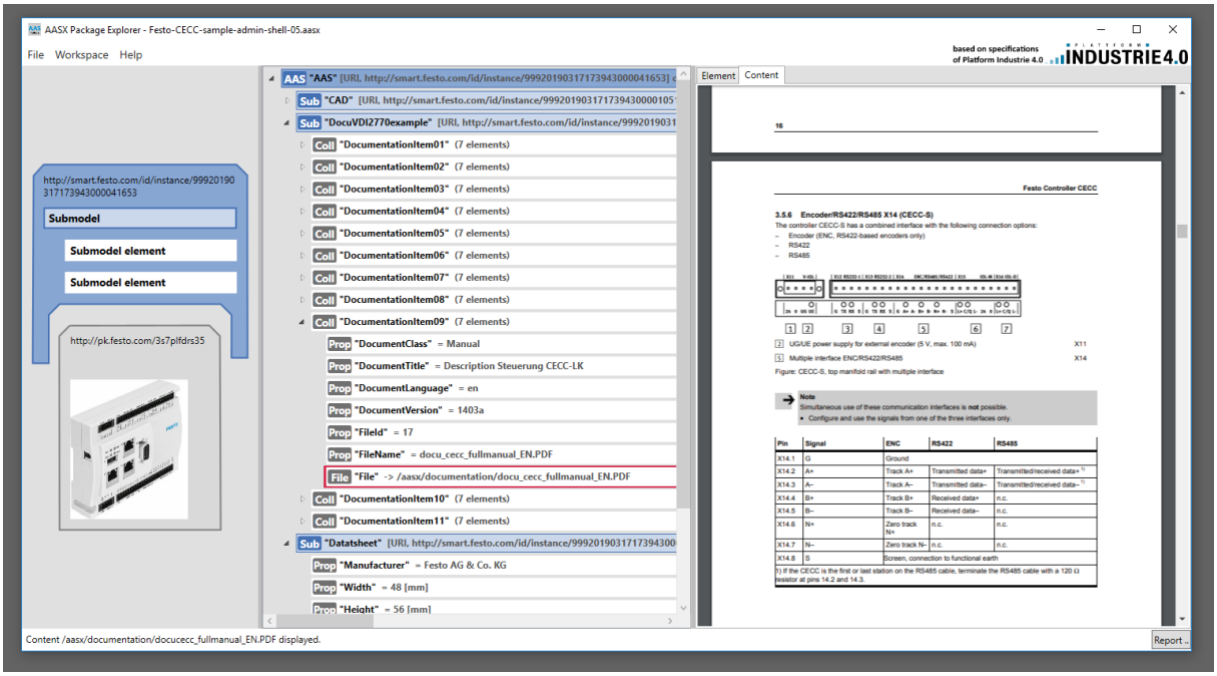
Figura 12 - Estrutura básica do AAS.



Fonte: Becker et al (BECKER, *et al.*, 2020).

O autor no artigo apresenta o software “AASX Package Explorer”, que é um editor e visualizador de AAS, como os da Figura 12 e sua interface na Figura 13. Assim, exportar os dados para diversos formatos serializados diferentes, por exemplo: XML, JSON e AML.

Figura 13 - Tela do editor e visualizador AASX Package Explorer.



Fonte: (AASX Package Explorer)

2.4.2. Modelo de Componente da Indústria 4.0

Outro modelo muito importante para a Indústria 4.0, que foi desenvolvido pela BITCOM, VDMA e ZWEI, é o modelo de componentes da Indústria 4.0, que se destina a ajudar os produtores e integradores de sistema a criar componentes de Hardware e Software para a Indústria 4.0. Ele permite uma melhor descrição das características ciberfísicas e permite a descrição da comunicação entre objetos e processos virtuais e ciberfísicos. Os componentes de Hardware e Software de produção serão capazes de cumprir as tarefas solicitadas por meio de recursos implementados especificados no modelo de componentes da Indústria 4.0 (ZEZULKA, *et al.*, 2016).

Os componentes da Indústria 4.0 complementam o trabalho da Plataforma Indústria 4.0 ao conectar produtos, equipamentos e processos, fazendo a ponte entre os padrões e as tecnologias da Indústria 4.0 no nível de produção. Um componente Indústria 4.0 é baseado nos chamados “objetos físicos”, que o termo inclui componentes, módulos, produtos, equipamentos como máquinas, ferramentas e fábricas e produtos como software (BURKE, 2018).

A característica mais importante é a capacidade de comunicação entre os objetos virtuais, processos com objetos reais e processos de produção, enquanto este modelo especifica a comunicação. A percepção física disso é que qualquer componente do sistema Industry 4.0 leva um contêiner eletrônico ou virtual (*shell*), de dados durante todo o ciclo de vida (ZEZULKA, *et al.*, 2016). O *shell* administrativo, ou do inglês Asset Administration Shell, é uma representação virtual do objeto físico e descreve suas funcionalidades. A comunicação é compatível com a Indústria 4.0 e ocorre por meio do *shell* administrativo do componente Indústria 4.0, internamente ou pela Internet. Através da conexão de produtos, equipamentos e processos com base na tecnologia da informação e comunicação, o mundo real está se aproximando cada vez mais da produção no mundo virtual da TI. Os objetos físicos e o *shell* administrativo reúnem-se para formar o componente Industry 4.0. Os produtos e equipamentos, projetados como componentes da Indústria 4.0, podem se comunicar uns com os outros, tanto dentro da fábrica quanto entre empresas (BURKE, 2018).

3- REVISÃO CONCEITUAL

Conceitos importantes e necessários sobre o RAMI 4.0 e OPC UA, abordados nesse trabalho, estão presentes de forma mais detalhada nessa seção.

3.1. RAMI 4.0

Empresas e instituições alemãs líderes do mercado, incluindo BITCOM, VDI/VDE e ZVEI (ZEZULKA, *et al.*, 2016), desenvolveram e lançaram em 2015 o RAMI 4.0, que é uma estrutura tridimensional. Este modelo de referência permite a migração passo a passo do mundo de hoje para a Indústria 4.0 e a definição de domínios de aplicação com convenções e requisitos especiais (ZEZULKA, *et al.*, 2018).

O RAMI 4.0 é um Modelo de Arquitetura de Referência para a Indústria 4.0 e foi desenvolvido para apoiar as iniciativas da Indústria 4.0, que estão alcançando ampla aceitação em todo o mundo. A Indústria 4.0 é uma visão integrada das empresas manufatureiras, iniciada na Alemanha, com esforços cooperativos em todo o mundo, incluindo China, Japão e Índia. Conceitos, estrutura e métodos da Indústria 4.0 estão sendo adotados em todo o mundo para modernizar a manufatura (LYDON, 2019).

O RAMI 4.0 também fornece uma visão de todos os aspectos importantes da Indústria 4.0 que são necessários para outras partes interessadas. Ele faz isso mapeando todos os aspectos críticos da Indústria 4.0 em três eixos diferentes, tornando-o uma espécie de mapeamento 3D das soluções da Indústria 4.0. Além disso, o RAMI 4.0 fornece orientações gerais e visões relevantes que devem ser levadas em consideração na organização geral do domínio, mas não prescreve uma abordagem de modelagem específica.

As seis camadas do eixo vertical definem a estrutura da representação de tecnologia da informação e comunicação, do termo em inglês *information and communication technology* (ICT), para um componente da Indústria 4.0. Este eixo vertical representa as várias perspectivas de ICT, tais como as aplicações de negócios (camada *Business*), os aspectos funcionais (camada *Functional*), o tratamento da informação (camada *Information*), a capacidade de comunicação e integração (camada *Integration*) e finalmente o hardware/ativos (camada *Asset*) envolvidos em um sistema ciber-físicos de produção (CPPS). O ciclo de vida de produtos, máquinas e fábricas são descritas ao longo do ciclo de vida e do eixo do fluxo de valor, do inglês

value stream. O terceiro eixo do RAMI 4.0 descreve a classificação funcional de várias circunstâncias dentro da Indústria 4.0, do produto ao nível da fábrica, até ao mundo conectado (SURI, *et al.*, 2017). A Figura 14 representa os eixos citados de forma gráfica.

Dada a integração, é possível que entidades atuantes na Indústria 4.0 busquem a comunicação mútua ao longo do ciclo de vida, independentemente das fronteiras entre empresas, nações e estados. Com isso, todas as entidades da cadeia produtiva poderão ter todos os dados necessários. Ressalta-se também que os participantes da cadeia produtiva e comercial, incluindo fabricantes de máquinas industriais e desenvolvedores de software, terão a oportunidade de desenvolver seus produtos com base no conhecimento prévio dos componentes mais recentes, ainda não desenvolvidos e testados em produção (ZEZULKA, *et al.*, 2018).

Adotar RAMI 4.0 como o padrão de design e aplicativos em Indústria 4.0 oferece alguns benefícios, como o uso de uma Arquitetura Orientada a Serviços (SOA), a combinação de componentes de TI em cada camada e a divisão de processos em componentes, facilitando a comunicação e processamento (MELO, 2020).

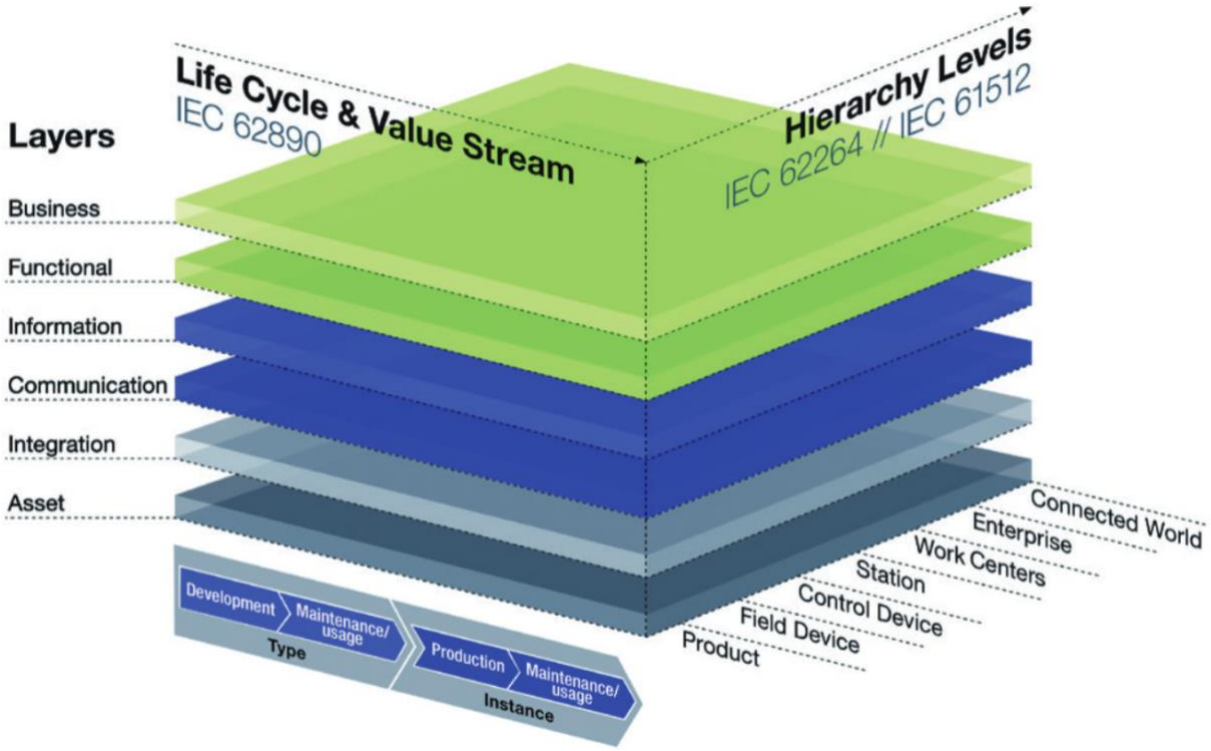
Diante disso, os trabalhos recentes têm se concentrado no desenvolvimento de soluções que integram diferentes tecnologias para implementar as funcionalidades exigidas para aplicações da Indústria 4.0, baseados em RAMI 4.0. Em Suri (2017), é proposto um método baseado em modelagem para criar e comunicar estratégias de negócios e fazer a ponte entre a estratégia de negócios e sua operação, usando o modelo *Business Motivation Model2* (BMM) do *Object Management Group* (OMG).

Para conseguir isso, foram criados modelos de BMM para comunicar ideias de negócios e conectar táticas de negócios correspondentes aos modelos de processos em BPMN, simular processos para verificar KPIs alcançáveis correspondentes a diferentes estratégias (modelados em BMM), e definir unidades organizacionais em BMM que executam tarefas modeladas em BPMN. Em Contrenas (2017), são abordadas as características essenciais que permitem que um sistema de manufatura seja adaptado para se tornar uma aplicação Indústria 4.0, seguindo o modelo RAMI 4.0. Para este fim, os autores especificam um sistema de manufatura inteligente baseado em padrões, como *Field Device Integration*, AutomationML e OPC UA.

O modelo de referência pode ser dividido em eixos estruturais RAMI 4.0, como mostrado na Figura 14 nos seguintes eixos: Níveis de hierarquia (Hierarchy Levels),

Ciclo de vida e fluxo de valor (Life Cycle & Value Stream) e Camadas ou Níveis (Layers).

Figura 14 - Modelo RAMI 4.0.



Fonte: Adolphs et al. (ADOLPHS, *et al.*, 2015).

3.1.1. Eixo de Níveis de Hierarquia

No eixo horizontal direito estão os níveis de hierarquia (*Hierarchy Levels*), do IEC 62264, a série de padrões internacionais para TI corporativa e sistemas de controle. Os níveis de hierarquia representam as diferentes funcionalidades dentro das fábricas ou instalações. Para representar o ambiente da Indústria 4.0, essas funcionalidades foram expandidas para incluir peças de trabalho, rotuladas como *Product* (Produto), *Field Device* (Dispositivos de campo), *Control Device* (Dispositivo de controle), *Station* (Estação), *Work Unit* (Unidades de trabalho), *Enterprise* (Negócio) e *Connected World* (Mundo conectado) (MARR, 2018).

Uma descrição mais precisa dos níveis de hierarquia é fornecida a seguir (ADOLPHS, *et al.*, 2015):

Produto: permite uma visão homogênea do produto a ser fabricado, da facilidade de produção e das interdependências entre eles.

Dispositivo de campo: representa o nível funcional de um dispositivo inteligente, por exemplo, um sensor inteligente. Basicamente, são dispositivos eletrônicos usados para detectar e identificar componentes e tecnologias de sensores.

Dispositivo de controle: considerado o cérebro da manufatura. Geralmente são máquinas/sensores usados para gerenciar comandos de entrada/saída, como Controlador Lógico Programável, da expressão em inglês *Programmable Logic Controller* (PLC), Sistemas de Controle Distribuído e Interface Gráfica.

Estação: onde os operadores realizam atividades administrativas para examinar o funcionamento de eventos e processos, como o uso do SCADA.

Unidades de trabalho: mantém informações de manufatura, define o estado de produção e supervisiona a renovação de matérias-primas para produtos refinados.

Negócio: é geralmente definido em termos de ERP, que é um software de gestão empresarial usados em processos de negócios, como planejamento de produção, entrega de serviços, marketing, vendas, módulos financeiros, varejo e outras despesas.

Mundo conectado: Este item foi adicionado na extremidade superior dos níveis de hierarquia para atender às necessidades da Indústria 4.0, que descrevem o grupo de fábrica e a colaboração com empresas externas de engenharia, fornecedores de componentes e clientes.

3.1.2. Eixo de Fluxo de Valor e Ciclo de Vida

No eixo horizontal esquerdo é representado o eixo do fluxo de valor do ciclo de vida de instalações e produtos, com base na IEC 62890. O gerenciamento do ciclo de vida para sistemas e produtos é usado na medição, controle e automação de processos industriais. Além disso, é feita uma distinção entre "tipos" (*type*) e "instâncias" (*instance*). Um tipo se torna uma instância quando o projeto e a prototipagem foram concluídos e o produto real está sendo fabricado, como podemos ver na base da Figura 14, caixas azuis. O modelo também combina todos os elementos e componentes de TI no modelo de camada e ciclo de vida (LYDON, 2019).

Conforme Adolphs (2015), um tipo é sempre criado com a ideia inicial. Isso inclui fazer pedidos de projetos, desenvolvimento e testes, desde a primeira amostra até a produção do protótipo (ADOLPHS, *et al.*, 2015). O tipo de produto, máquina etc, podem ser criados durante esta fase. Os autores ressaltam ainda que os produtos são

fabricados industrialmente com base no tipo geral, sendo que cada produto fabricado representa essa instância, que pode ter, por exemplo, um número de série único.

3.1.3. Eixo de Camadas

O eixo das camadas ou níveis no eixo vertical, descreve a decomposição de uma máquina em suas propriedades, estruturadas por camada, o mapeamento virtual de uma máquina. Tais representações se originam da tecnologia da informação e comunicação, onde propriedades de sistemas complexos são comumente divididas em camadas (LYDON, 2019).

O eixo das camadas ou níveis do RAMI 4.0 pode ser descrito pelos seguintes aspectos e partes:

Camada de Regras de Negócios (*Business Rules Layer*): abrange modelos de negócios abstratos e a lógica de processo, bem como garante a integridade das funções ao longo da cadeia de valor (WANG; TOWARA; ANDERL, 2017). A camada garante a integridade das funções no fluxo de valor, permite o mapeamento de modelos de negócios e o resultado do processo geral. Contém as regras legais e de regulamentação, permitindo modelar as regras que o sistema deve seguir. A camada também cria uma ligação entre diferentes processos de negócios (ZEZULKA, *et al.*, 2016):

Camada funcional (*Functional Layer*): permite uma descrição formal de regras e cria uma plataforma para combinar horizontalmente as várias regras ou funções. Contém processos de execução codificados, ambiente de modelagem para serviços de suporte a processos de negócios, ambiente de execução para aplicativos e funcionalidade técnica. As regras e a lógica de tomada de decisão são geradas na camada Funcional. Alguns casos de uso também podem ser executados nas camadas inferiores. Porém, o acesso remoto e a integração horizontal podem ocorrer apenas dentro da camada Funcional e devido à necessidade de integridade dos dados (ZEZULKA, *et al.*, 2016).

Camada de informação (*Information Layer*): fornece processamento e execução para pré-processamento de eventos e execução de regras relacionadas a eventos. A camada permite a descrição formal das regras e o pré-processamento de eventos (ZEZULKA, *et al.*, 2016). Para isso, os dados são verificados quanto à sua integridade, resumidos em dados novos com maior qualidade e disponibilizados para

camadas superiores, como Funcional (WANG; TOWARA; ANDERL, 2017). Ele também recebe eventos e os transforma para corresponder aos dados que estão disponíveis para a camada superior. (ZEZULKA, *et al.*, 2016).

Camada de Comunicação (*Communication Layer*): esta camada fornece a padronização da comunicação por meio de formato uniforme de dados na direção da Camada de Informação (ZEZULKA, *et al.*, 2016). Esta camada fornece serviços para controlar a camada de integração (WANG; TOWARA; ANDERL, 2017).

Camada de Integração (Integration Layer): fornecendo informações relacionadas a ativos técnicos, como hardware e software, para as camadas acima. Esta camada contém todos os elementos associados à TI, incluindo a interface homem-máquina, do inglês *human-machine interface* (HMI), gerando eventos com base nas informações adquiridas e realizando o controle final do processo (WANG; TOWARA; ANDERL, 2017). Esta camada fornece informações sobre os ativos (Hardware/Software e componentes), em um formulário que está disponível para processamento informatizado. Também, é responsável pelo controle informatizado do processo, geração de eventos a partir dos ativos e contém elementos, que estão conectados à TI, como leitores RFID, sensores, HMI e atuadores. A integração de pessoas também faz parte das funções da camada de integração, via HMI (ZEZULKA, *et al.*, 2016).

Camada de Ativos (Asset Layer): esta é a camada mais baixa na estrutura RAMI 4.0. Representa a parte física, contendo objetos como processos, máquinas, sensores, atuadores e documentação. Esta camada identifica produtos e peças em relação aos requisitos para as máquinas e estações de trabalho para processamento (WANG; TOWARA; ANDERL, 2017). Humanos fazem parte da Camada de Ativos, por estarem conectados ao mundo da realidade virtual pela camada de Integração. A conexão passiva dos ativos à camada de Integração superior é feita, por exemplo, por meio de QR *codes* (ZEZULKA, *et al.*, 2016).

Os três eixos do modelo RAMI 4.0 possuem os aspectos cruciais para implementação da Indústria 4.0 e podem ser mapeados, permitindo que objetos como máquinas sejam classificados de acordo com o modelo. Conceitos altamente flexíveis da Indústria 4.0 podem ser descritos e implementados usando RAMI 4.0. O modelo permite a migração passo a passo do presente para o mundo da Indústria 4.0 (LYDON, 2019).

3.1.4. Comunicação Aberta para a Indústria 4.0

A Indústria 4.0 é impulsionada por tecnologias de informação e comunicação avançadas (ZEZULKA, *et al.*, 2018). As discussões iniciais, sobre a visão da Indústria 4.0, estão cada vez mais evoluindo para soluções específicas para empresas, que vão desde protótipos conceituais até a produção conectada. No entanto, não existe uma solução única e perfeita neste contexto. Em vez disso, é uma questão de como cada empresa interpreta a Indústria 4.0, já que cada empresa precisa pensar sobre quais benefícios ela ganhará com a Indústria 4.0. Porém, as soluções da Indústria 4.0 só podem ser implementadas com sucesso, se houver comunicação aberta entre as empresas (MARSEU; KOLBERG; WEYER, 2017).

Conectar o conhecimento e capacidade da empresa é a chave para o sucesso, já que a Indústria 4.0 só pode ganhar vida, se as empresas estiverem conectadas em um nível tecnológico, a conexão inteligente é a chave. Esse potencial inclui a produção específica para o cliente e a transição para novos modelos de negócios ao longo de todo o ciclo de vida do produto, desde a concepção até o descarte (MARSEU; KOLBERG; WEYER, 2017). Por conta dessas solicitações, para atendê-las, contamos com o protocolo OPC UA, que é extremamente conveniente para a infraestrutura de informação e comunicação do Indústria 4.0, pois permite comunicações padronizadas, rápidas e seguras (ZEZULKA, *et al.*, 2018). Em (HOPPE, 2017), fica claro que não há possibilidade da Indústria 4.0 sem OPC UA. O RAMI 4.0 recomendou apenas OPC UA para a implementação da funcionalidade da camada de comunicação sobre outras tecnologias existentes para Indústria 4.0 e IIoT. Portanto, (ZEZULKA, *et al.*, 2018), destaca que OPC UA será aceito como um protocolo de comunicação comum para a revolução Indústria 4.0 nos países industriais mais desenvolvidos em breve.

3.2. OPC UA

OPC UA é o padrão de troca de dados para comunicação industrial segura, confiável e independentemente da plataforma e fabricante. Ele permite a troca de dados entre produtos de diferentes fabricantes e entre diferentes sistemas operacionais. O padrão OPC UA é baseado em especificações que foram desenvolvidas em cooperação de fabricantes, usuários, institutos de pesquisa e consórcios. Assim, permitindo a troca segura de informações em sistemas heterogêneos (BURKE, 2018). Em (KHAKIFIROOZ, *et al.*, 2018), com a última

revolução industrial a caminho, o OPC UA se revela como o componente fundamental para suportar a interconectividade e interoperabilidade em todos os níveis.

O padrão foi criado e mantido pela OPC Foundation, para que o OPC UA seja um padrão IEC e, portanto, ideal para cooperação com outras organizações. A OPC Foundation coordena o desenvolvimento do padrão OPC em colaboração com usuários, fabricantes e pesquisadores, como uma organização global sem fins lucrativos (BURKE, 2018).

3.2.1. OPC Clássico

A OPC Foundation iniciou suas atividades em 1994. Em 1995, os fornecedores de automação Fisher-Rosemount, Intellution, Opto 22 e Rockwell Software, formam uma força-tarefa para desenvolver um padrão de acesso a dados baseado em COM (*Component Object Model*) e DCOM (*Distributed COM*), e foi dado o nome de OPC. OPC sendo uma abreviatura de OLE (*Microsoft Object Linking & Embedding*), para controle de processos (History - OPC Foundation, 2019). Hoje em dia, essa primeira versão é chamada de OPC clássico. O próprio protocolo define a forma como o OPC-Client e o OPC-Server se comunicam e trocam informações. As especificações fornecem definições separadas para acessar dados de processo, alarmes e dados históricos (Classic - OPC Foundation, 2019).

Conforme descrito por Wolfgang et al. (2009), na época em que o OPC foi estabelecido, a fundação OPC reduziu o trabalho de especificação para definir diferentes APIs sem a necessidade de definir o protocolo de rede ou mecanismo de intercomunicação. Usando tecnologias disponíveis em todos os PCs baseados em sistemas operacionais Windows, a redução no trabalho levou ao rápido desenvolvimento das especificações, o que resultou na redução do tempo de colocação no mercado dos produtos OPC disponíveis.

Seguem os padrões e especificações do OPC clássico:

- ✓ OPC *Data Access* (OPC DA), a especificação OPC DA define a troca de dados incluindo valores, tempo e informação de qualidade (Classic - OPC Foundation, 2019).
- ✓ OPC *Alarms & Events* (OPC AE), a especificação OPC AE define a troca de informações de mensagens de alarmes e eventos, bem como estados variáveis e gerenciamento de estados (Classic - OPC Foundation, 2019).

- ✓ OPC *Historical Data Access* (OPC HDA), a especificação OPC HDA define métodos de consulta e análises que podem ser aplicados a dados históricos com data especificada (Classic - OPC Foundation, 2019).

3.2.2. Conceitos do OPC UA

Em 2008, a OPC Foundation lançou a *OPC Unified Architecture* (OPC UA), uma arquitetura orientada a serviços independente de plataforma, que integra todas as funcionalidades das especificações existentes no OPC clássico e continua compatível com as versões anteriores do OPC Classic. Vários fatores influenciaram a decisão de criar OPC UA (Classic - OPC Foundation, 2019). Essas decisões estão relacionadas à interoperabilidade de dados, segurança e ter uma arquitetura multiplataforma SOA (*Service-Oriented Architecture*). O OPC UA é um padrão amplamente reconhecido em automação industrial, para interoperabilidade e troca de dados nos níveis de sensores e atuadores, por meio de sistemas de controle e comunicação para servidores centrais e nuvens, que desempenha um papel fundamental neste processo (DRAHOš, *et al.*, 2018).

As principais vantagens do OPC UA são:

Modo de acesso a dados unificado: o OPC tradicional é independente para diferentes aplicações. Ao contrário, o OPC UA Server integra dados, alarmes e eventos em um único espaço de endereço. Os clientes podem obter aplicativos de dados, alarmes e eventos apenas por meio de uma chamada de método (WOLFGANG; STEFAN-HELMUT; MATTHIAS, 2009).

Suporta estruturas de dados complexas: o OPC UA oferece modelos de meta informação que podem ser facilmente estendidos. No OPC UA, você pode adicionar e excluir as ligações entre esses modelos de dados. Portanto, não é necessário o cliente entender o significado dos dados com as descrições detalhada do modelo. Esta medida torna mais fácil desenvolver o software cliente e melhora muito a precisão da significância dos dados (LU; ZHIFENG, 2010).

Suporte a várias plataformas: as especificações OPC tradicionais são baseadas na tecnologia Microsoft COM/DCOM. Com o desenvolvimento do .NET e dos serviços Web, a Microsoft não se concentrou mais na tecnologia COM. Além disso, os fornecedores precisam de uma especificação independente de plataforma ou sistema operacional e não somente em um sistema baseado em Windows. OPC

Foundation oferece três SDKs diferentes com suporte às linguagens de programação C, C++, C# e Java. Você pode desenvolver OPC UA em Windows, Linux e dispositivos embarcados (LU; ZHIFENG, 2010).

Mecanismos de segurança aprimorados: o OPC tradicional não possui projeto de segurança próprio e depende totalmente da segurança do COM/DCOM. No entanto, OPC UA define um conjunto de mecanismos de segurança abrangente. Entre o canal de segurança do cliente e do servidor, ele possui *handshake* bidirecional para autenticar os certificados IEEE 279. O OPC UA usou principalmente a tecnologia de segurança, com a infraestrutura de chave pública e certificados X509v3 (LU; ZHIFENG, 2010).

3.2.3. OPC UA e RAMI 4.0

De uma perspectiva de TI, OPC UA é a interface de programação para a "fábrica conectada" e qualquer outra instalação industrial, e é um ativador crítico para a Internet das Coisas Industrial (IIoT), bem como a adoção do RAMI4.0 (BURKE, 2018). OPC UA é um protocolo para comunicação industrial e foi padronizado na IEC 62541 (GRÜNER; PFROMMER; PALM, 2016). A Figura 14 nos mostra um mapeamento dos protocolos OPC UA no modelo geral RAMI 4.0 principal (ZEZULKA, *et al.*, 2018).

A abordagem para implementação de uma camada de comunicação é feita pelo padrão básico IEC 62541 (arquitetura unificada OPC, 2017; ZEZULKA, *et al.*, 2018). A abordagem para implementação de uma camada de informação (RAMI 4.0), segue a norma IEC *Common Data Dictionary* (IEC 61360 Series/ISO 13584-42); características, classificação e ferramentas para eCl@ss; *Electronic Device Description* (EDD); *Field Device Tools* (FDT) (ZEZULKA, *et al.*, 2018). A abordagem para implementação da camada da informação por *Field Device Integration* (FDI), como tecnologia de integração (ZEZULKA, *et al.*, 2018). A abordagem para engenharia de ponta a ponta com o uso do Automation ML (ZEZULKA, *et al.*, 2018).

A Arquitetura Unificada OPC (UA), lançada em 2008, é uma arquitetura orientada a serviços independente de plataforma, que integra todas as funcionalidades das especificações OPC Classic em uma estrutura extensível (Unified Architecture - OPC Foundation, 2019).

A abordagem em várias camadas cumpre os objetivos da especificação de projeto original de (Unified Architecture - OPC Foundation, 2019):

Equivalência funcional: todas as especificações COM OPC Classic são mapeadas para UA;

Independência de plataforma: compatível desde um microcontrolador integrado até uma infraestrutura baseada em nuvem;

Seguro: criptografia, autenticação e auditoria;

Extensível: capacidade de adicionar novos recursos sem afetar os aplicativos existentes;

Modelagem de informações abrangente: para definir informações complexas. O OPC UA pode ser usado em muitos componentes nos domínios industriais para troca de informações entre sensores, sistemas de controle, atuadores, sistemas de manufatura e sistemas de planejamento. O OPC UA melhorou o suporte de modelagem da informação em relação ao OPC clássico (Unified Architecture - OPC Foundation, 2019):

Discovery: encontra a disponibilidade de Servidores OPC em PCs e/ou redes locais;

Espaço de endereçamento: todos os dados são representados hierarquicamente (por exemplo, arquivos e pastas), permitindo que estruturas simples e complexas sejam descobertas e utilizadas pelos Clientes OPC;

Sob demanda: ler e gravar dados/informações com base nas permissões de acesso;

Assinaturas (Subscriptions): monitorar dados/informações e relatório por exceção quando os valores mudam com base nos critérios de um cliente;

Eventos: notificar informações importantes com base nos critérios do cliente;

Métodos: os clientes podem executar programas etc, com base em métodos definidos no servidor;

Plataformas de hardware: pode ser executado em hardware tradicional de PC, servidores baseados em nuvem, PLCs e microcontroladores (ARM etc.);

Sistemas operacionais: Microsoft Windows, Apple OSX, Android ou qualquer distribuição de Linux etc.

Os *OPC Base Services* são descrições abstratas de métodos, independentes de protocolo e fornecem a base para a funcionalidade do OPC UA. A camada de transporte coloca esses métodos em um protocolo, o que significa que ela serializa os

dados e os transmite pela rede. Dois protocolos são especificados para este propósito. Um é um protocolo TCP binário, otimizado para alto desempenho e o segundo é orientado a serviços da Web (ZEZULKA, *et al.*, 2018).

A visão da OPC Foundation é oferecer uma plataforma de múltiplos domínios para interoperabilidade e troca de dados de sensores e atuadores para gerenciamento de sistemas corporativos. OPC UA é mais do que apenas um protocolo. OPC UA é um framework para a representação e troca de dados e informações orientados a objetos (DRAHOš, *et al.*, 2018).

Para troca de informações, o OPC UA oferece dois mecanismos de comunicação:

O primeiro é um modelo cliente-servidor, onde o cliente acessa as informações fornecidas pelo servidor por meio de serviços definidos (DRAHOš, *et al.*, 2018). Webservices (HTTP / HTTPS / SOA) e CoAP são protocolos que implementam um modelo cliente-servidor seguindo as descrições dos OPC *Base Services*. Em geral, as arquiteturas servidor/cliente usam comunicação orientada à conexão. Os clientes estabelecem conexões com os servidores. O cliente usa chamadas de serviço sobre essas sessões para o servidor. Manter essas sessões, em um cenário um-para-muitos, pode alocar muitos recursos em seu dispositivo, o que pode ser um problema, especialmente para dispositivos embarcados com poucos recursos de hardware. Este problema pode ser resolvido usando OPC UA PubSub (ECKHARDT; MULLER; LEURS, 2018).

O segundo método de comunicação é o OPC UA PubSub, que está desde aproximadamente 2017 em fase de testes (DRAHOš, *et al.*, 2018). O OPC UA PubSub pode ter um conceito *Broker*, *Brokerless* ou Transporte.

Conceito de Broker: é um middleware entre duas entidades que desejam trocar dados. O remetente dos dados, chamado editor (*publisher*), publica suas mensagens nos tópicos de um *Broker*. O receptor, chamado de assinante (*subscriber*), pode notificar o *Broker* de que está interessado em um tópico. Posteriormente, o assinante recebe as mensagens do *Broker*. Um *Broker* pode desacoplar a comunicação entre as entidades no espaço, tempo e sincronização (ECKHARDT; MULLER; LEURS, 2018).

3.2.4. Segurança do OPC UA

Uma das considerações mais importantes na escolha de uma tecnologia é a questão da segurança (Unified Architecture - OPC Foundation, 2019). O OPC tradicional não possui um projeto de segurança próprio e depende totalmente da segurança do COM/DCOM. No entanto, OPC UA define um conjunto de mecanismos de segurança abrangente. Entre o canal de segurança proprietário do cliente e do servidor, ele tem um *handshake* bidirecional para autenticar os dois certificados como foi observado em trabalhos de (LU; ZHIFENG, 2010) e (GRÜNER; PFROMMER; PALM, 2016).

OPC UA interage bem com o firewall enquanto aborda questões de segurança, fornecendo um conjunto de controles, como (Unified Architecture - OPC Foundation, 2019):

Transporte: vários protocolos são definidos fornecendo opções como o transporte binário OPC ultrarrápido ou o SOAP-HTTPS mais universalmente compatível, por exemplo.

Criptografia de sessão: as mensagens são transmitidas com segurança em níveis de criptografia de 128 ou 256 bits;

Assinatura de mensagens: as mensagens são recebidas exatamente como foram enviadas;

Pacotes sequenciados: a exposição a ataques de repetição de mensagem é eliminada com o sequenciamento;

Autenticação: cada cliente e servidor UA é identificado por meio de certificados OpenSSL, fornecendo controle sobre quais aplicativos e sistemas têm permissão para se conectar entre si;

Controle do usuário: os aplicativos podem exigir que os usuários se autentiquem (credenciais de login, certificado etc), e podem restringir e aprimorar ainda mais suas capacidades com direitos de acesso e “visualizações” de espaço de endereço;

Auditoria: atividades por usuário e/ou sistema são registradas fornecendo um rastreio de registros para fins de auditoria de acesso.

4- MATERIAIS E MÉTODOS

A seguir serão apresentados conceitos importantes para a elaboração do trabalho, e as tecnologias escolhidas para executar a prova de conceito.

4.1. Proposta do Trabalho

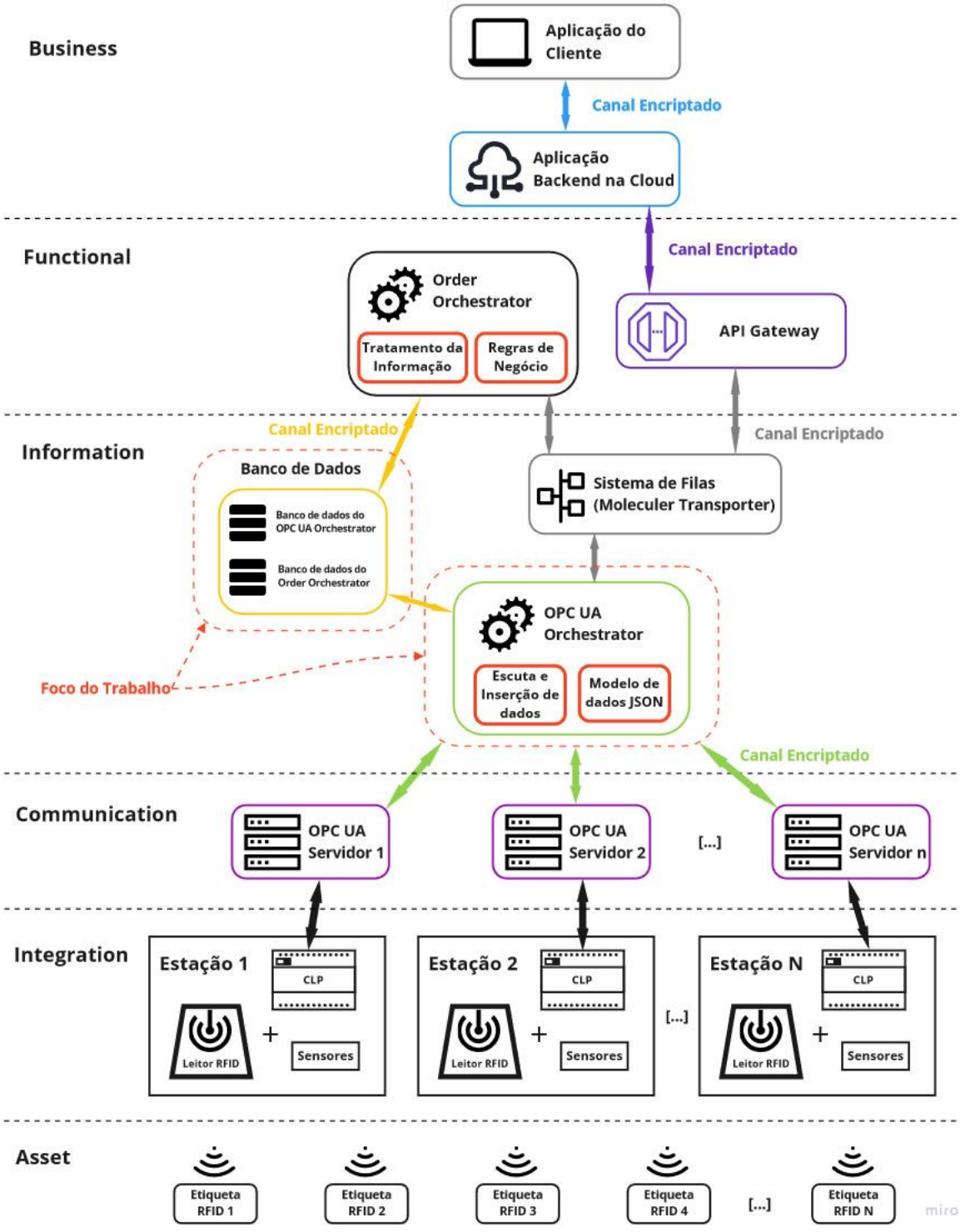
Este trabalho faz parte de um projeto de pesquisa que visa desenvolver uma arquitetura orientada a microsserviços baseado no RAMI 4.0, para aplicações de manufatura flexível na Indústria 4.0, e focará nos desenvolvimentos e discussões relativas à camada da Informação do RAMI 4.0. Uma visão geral macro das camadas da arquitetura propostas nesse projeto de pesquisa são apresentadas na Figura 15. É importante observar que esta dissertação contempla o desenvolvimento da parte relacionada à camada de Informação na Figura 15, o qual resumidamente considera o elemento *OPC UA Orchestrator* e a modelagem da Informação utilizando o formato JSON e com banco de dados não relacional.

Observe que a arquitetura descreve desde a *tag* RFID em ambiente de chão de fábrica (camada *Asset*), seguindo pela estação de leitura, onde os dados da *tag* RFID são lidos, interpretados e enviados para a o tratamento no OPC UA. As estações RFID são as estações de manufatura (camada *Integration*), responsáveis pelo controle lógico de produção e leitura das *tags* através da identificação de radiofrequência. A comunicação dessas estações com o OPC UA é bidirecional, visto que ela está sempre enviando informações que chegaram até ela e questionando a rede o que deve ser feito com aquele item (camada *Communication*). As partes relacionadas às camadas de *Asset*, *Integration* e *Communication* foram consideradas em trabalhos anteriores (MELO, *et al.*, 2021) e (MELO, 2020).

O padrão OPC UA é descrito através do(s) servidor(es) e do orquestrador. O servidor OPC UA é o bloco de captura e envio de informações nas estações de manufatura. Ele é responsável por saber de onde cada informação saiu e para onde ela deve ir através da estrutura de tópicos do OPC UA. O Orquestrador (*orchestrator*) OPC UA é responsável pela recepção dessas informações e pela tomada de decisão, baseado em regras de negócio. Ou seja, enquanto o servidor OPC UA conhece o fluxo da informação nas camadas inferiores, o OPC UA *Orchestrator* conhece o fluxo da informação nas camadas superiores, baseado em regras de negócio e regras de tomada de decisão. Toda a tomada de decisão é baseada nas informações que vêm

da fábrica e nas informações contidas no banco de dados, que além de regras, traz a visão de produção e histórico do registro das *tags* ou etiquetas. Toda a informação trafegada aqui é criptografada por certificados seguros.

Figura 15 - Visão macro das camadas do RAMI 4.0 aplicados na proposta.



Fonte: Autoria própria.

O OPC UA *Orchestrator* e a modelagem da informação no banco de dados estão relacionados à camada da Informação (*Information*). O banco de dados, estruturado em linguagem NoSQL e temporal, ou seja, não estruturada e com a inserção da data e hora de quando cada informação foi inserida. Com tabelas específicas do banco de dados que serão vistos em breve, toda a informação do chão de fábrica e armazenada durante a fabricação, permitindo um acompanhamento em tempo real da ordem de produção requisitada e da ordem de produção sendo fabricada.

O Orquestrador *Order* e a API *Gateway* estão relacionados à camada funcional (*Functional layer*). Esses elementos estão sendo desenvolvidos em outro trabalho do mesmo grupo de pesquisa (NETO, 2022). O *Order Orchestrator* é responsável por trazer dados das camadas de Negócios para a camada Integração. Em outras palavras, refere-se ao desejo de fabricação do cliente em relação à realidade de produção. É nesta camada que novas ordens são inseridas na produção, o status de fabricação é monitorado e os desejos de atualização de produção são considerados viáveis ou não para habilitação da capacidade de reconfigurar ou flexibilizar a produção.

Para torná-lo mais abrangente, o *Order Orchestrator* foi separado em três blocos principais que fazem a recepção de pedidos, aplicação de regras comerciais, priorização e parametrização da ordem no banco de dados. A comunicação é feita por meio do API Gateway que leva essas informações de forma organizada e estruturada para a interface do cliente. Para elevar a segurança, as informações são criptografadas via protocolo HTTPS, autenticação e autorização com o protocolo OAuth 2.0. Usando *Transport Security Layer* (TLS), tanto o OPC UA *Orchestrator*, quando o *Order Orchestrator* conversam com o banco de dados, encriptando o canal de comunicação.

Todos os clientes são identificados no banco de dados, e uma Ordem/Pedido específica está sempre relacionada a um cliente. Um cliente tem uma integração baseada em API usando o API Gateway, para que possa colocar novos Pedidos, seguir o status de fabricação do Pedido e modificar ou excluir ordens em andamento.

Aplicativos de *backend* na nuvem são responsáveis pela orquestração do cliente e da integração interna. Essa parte está relacionada à camada de Negócios (*Business*). Do ponto de vista da fábrica, o *Order Orchestrator* é responsável por organizar, manusear e aplicar regras de negócios às informações do lado do cliente e

da fábrica. Isso é feito para garantir que a produção seja corretamente seguida e centralizada no banco de dados.

A perspectiva de negócio da arquitetura consiste na integração, padronização e comunicação do processo de produção. Qualquer cliente, seja um ou milhares deles, fabricando um ou milhares de itens, tem sua interface padronizada através das APIs RESTful. A aplicação do cliente é vista como a plataforma de integração que faz sentido ao seu negócio, seja ela via aplicativo de celular, website, ou via software dedicado (ERP). Por meio das APIs, qualquer interface pode ser conectada a fábrica, enviando e consumindo informações dela.

4.2. Tecnologias Utilizadas

A seguir serão apresentadas as principais tecnologias que foram fundamentais para a concepção da proposta e uma breve descrição sobre cada um deles, de modo a deixar a proposta mais clara ao leitor.

4.2.1. Node.js

Os sistemas .NET, Java, Python possuem em comum o bloqueio de I/O no servidor ou *blocking-thread*. Em um servidor web pode-se visualizá-lo de forma ampla, considerando que cada processo é uma requisição feita pelo usuário. Com o aumento de usuários acessando e gerando uma requisição no servidor, o sistema enfileira cada requisição e depois as processa, uma a uma. Enquanto uma requisição é processada as demais ficam em espera, mantendo por um período uma fila de requisições ociosas (PEREIRA, 2014).

Node.js é uma plataforma altamente escalável e de baixo nível, visto que a programação é diretamente com os protocolos de rede e internet ou utilizando bibliotecas que acessam recursos do sistema operacional. O Javascript é a linguagem base de programação, e foi possível através do motor Javascript V8 do Google Chrome. (CANTELON, *et al.*, 2013)

As aplicações são *single-thread*, onde cada aplicação terá uma única instância e um único processo. Não é possível trabalhar com programação concorrente em plataforma *multi-thread*. Para utilizar ao máximo da programação assíncrona, podem-se utilizar as funções em background que aguardam o seu retorno através de funções de *callback* e tudo isso é trabalhado de forma não-bloqueante (PEREIRA, 2014).

O Node.js é uma linguagem orientada a eventos. Ou melhor, a orientação é o trabalho com eventos de I/O do servidor, exemplo: o evento *connect* de um banco de dados, um *open* de um arquivo, uma data de um *streaming* de dado, entre outros. Event-Loop é o *trigger*, que fica escutando alguma coisa e lançando eventos de execução para os módulos. Na prática é um *loop* infinito que a cada iteração verifica em uma fila de eventos se algum deles foi emitido. Quando ocorre, é emitido um evento e enviado para uma fila de execução, podendo-se programar e executar novas ações em uma função de *callback*.

A plataforma Node.js apresenta algumas características importantes, como (KOTA; PRABHU, 2013):

1. Execução assíncrona: modelo de evento mais adequado para a criação de aplicativos Web, altamente escalonáveis por meio de retornos de chamada;
2. Menor curva de aprendizado: desenvolvedores estão familiarizados com JavaScript e programação assíncrona;
3. Praticidade: avanços na velocidade de execução tornaram mais objetivo escrever software do lado do servidor;
4. Compartilhamento de código: o desenvolvimento em cliente e servidor pode ser reaproveitado, o que ajuda a reduzir a alternância de contexto entre o desenvolvimento de cliente e servidor, permitindo o compartilhamento de código;
5. NoSQL: JavaScript é a linguagem usada em vários bancos de dados, como, por exemplo, CouchDB, MongoDB;
6. JSON: é um formato de dados simples, eficiente e popular nos dias de hoje.

4.2.2. Framework Moleculer

Moleculer é um projeto de código aberto, com uma estrutura de microsserviço moderna, rápida e poderosa para Node.js. Seu potencial está na criação de serviços confiáveis e escaláveis. O Moleculer fornece muitos recursos para criar e gerenciar seus microsserviços. Seu *framework* apresenta uma estrutura base ou uma plataforma de desenvolvimento, que contém ferramentas, bibliotecas, sistemas e componentes que agilizam o processo de desenvolvimento de uma solução específica (MOLECULERJS, 2021).

Uma vantagem do Moleculer é que todos os microsserviços desenvolvidos

possuem disponível um recurso automático de registro e descoberta, ou seja, todos os serviços existentes são informados após a criação de um novo serviço ou a disponibilização de uma nova funcionalidade. Outro recurso é o balanceamento automático de carga, o qual tem função de distribuir dinamicamente a carga da comunicação entre os microsserviços uniformemente (PONTAROLLI, 2020).

O serviço *Transporter* é um dos componentes mais importantes da arquitetura pois é o mecanismo que coordena o envio e a recepção de mensagens em uma fila. O Transporter é capaz de enfileirar e manter as mensagens até serem processadas por consumidores. O produtor simplesmente envia a mensagem para o Transporter, sem a necessidade de um mecanismo de descoberta de serviços. No Moleculer, o serviço *Transporter* utiliza um protocolo de comunicação para mensageria, como TCP, NATS e AMQP, ou seja, para se comunicar com os outros microsserviços. O grande benefício dessa abordagem é o desacoplamento entre produtores e consumidores de eventos e dados (PONTAROLLI, 2020).

O serviço de mensageria ou *Transporter* é uma técnica que busca solucionar a comunicação entre dispositivos e sistemas diferentes de uma maneira confiável. O *broker* da mensagem faz com que seja possível integrar tais sistemas para que eles possam trocar dados de forma desacoplada. De acordo com (BIGHETI, 2020), citando também a documentação oficial do Moleculer (MOLECULERJS, 2021), os principais recursos para microsserviços que compõem o framework Moleculer são:

1. Conceito de requisição-resposta;
2. Arquitetura baseada em eventos;
3. Suporte a middlewares para Node.js;
4. Suporte ao armazenamento em cache de variável;
5. Diversas opções de comunicação (*Transporters*);
6. Diversas opções de serializers: JSON, MsgPack, Protocol Buffer etc;
7. Suporte ao desenvolvimento de múltiplos serviços em um mesmo nó;
8. Suporte nativo para o registro de serviços;
9. Descoberta automática de serviços;
10. API para interface com aplicações externas.

O Moleculer disponibiliza uma série de mecanismos diferentes para comunicação entre os microsserviços como: TCP/IP, NATS, MQTT, AMQP e Kafka. O Transportador ou *Transporter* é um módulo importante para executar serviços em

vários nós, já que o transportador se comunica com todos eles. A comunicação se dá na transferência de eventos, solicitações de chamadas e respostas de processos. Se um serviço for executado em várias instâncias em nós diferentes, as solicitações terão balanceamento de carga entre os nós ativos (MOLECULER TRANSPORTER, 2021). Foi escolhido o protocolo AMQP como o *Transporter* para o *framework* Moleculer nesse projeto, que usa o software RabbitMQ como base.

4.2.3. Node-RED e node-opcua

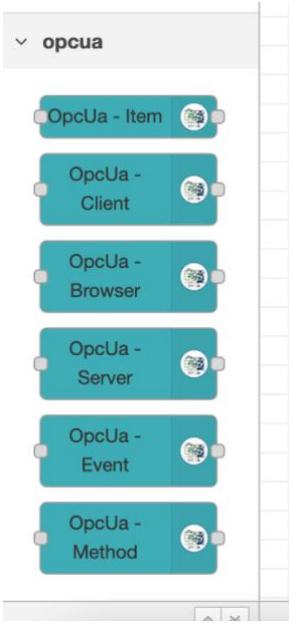
Apesar desse trabalho não ter o foco na camada de Comunicação, será necessário acessar dados presentes nesta camada por meio de um ou mais servidores OPC UA. Assim sendo, os conceitos explicados na seção 3.2, tornaram-se necessários para realizar a implementação de software OPC UA dentro da camada de comunicação.

A implementação de software escolhida foi o Node OPC UA junto com a ferramenta de programação RAD chamada Node-RED. O Node-RED é utilizado para facilitar o desenvolvimento com suporte a diagramas de fluxo, interface WEB e a disponibilidade de vários *plugins* Node.js.

O plugin node-opcua utiliza a licença MIT e traz a implementação de um servidor OPC UA escrita na liguagem JavaScript e Node.js para sua execução. O Node.js foi selecionado, pois esta linguagem é uma excelente escolha para aplicações orientadas ao servidor, com várias vantagens como, fácil escalabilidade, manipulação de requisições simultaneamente e altamente extensível (MALVI, 2018).

Na Figura 16, pode-se ver a lista de componentes do *plugin* node-opcua para Node-RED, onde temos os nodes importantes, como: o “OpcUa - Server”, que é a caixa para criar um servidor OPC UA e o “OpcUa - Item”, onde são configuradas as variáveis de entrada e saída e suas nomenclaturas.

Figura 16 - Funcionalidades disponíveis Node-RED / node-opcua.



Fonte: Autoria própria.

4.2.4. MongoDB

Segundo (CHAUHAN, 2019) o MongoDB é o mais popular entre os bancos de dados NoSQL, além de ser uma ótima ferramenta para a construção de *data warehouses*, especialmente por conta da sua capacidade de utilizar a arquitetura de *cluster* sem fragmentação. É um banco de dados de código aberto, o que o torna ideal para a construção de data *warehouses* de alto desempenho.

MongoDB oferece suporte a operações autônomas ou de instância única. Os conjuntos de réplicas fornecem alto desempenho de replicação com tratamento automatizado de falhas, enquanto os *clusters* fragmentados tornam possível dividir grandes conjuntos de dados em diferentes máquinas que são transparentes para os usuários. Os usuários do MongoDB combinam conjuntos de réplicas e *clusters* fragmentados, para fornecer altos níveis de redundância de conjuntos de dados, que são transparentes para implementações de softwares (DAMODARAN, 2016).

Ao projetar o esquema de um banco de dados, é impossível saber com antecedência todas as consultas que serão realizadas pelos usuários finais. Uma consulta *ad-hoc* é um comando de curta duração cujo valor depende de uma variável. Cada vez que uma consulta *ad-hoc* é executada, o resultado pode ser diferente, dependendo das variáveis em questão. O MongoDB oferece suporte a consultas de campo, consultas de intervalo e pesquisas de expressão regular. As consultas podem

retornar campos específicos e contabilizar funções definidas pelo usuário. Isso é possível porque o MongoDB indexa documentos BSON e usa o MongoDB Query Language (MQL) (MongoDB Features, 2021).

Em respeito a indexação de dados e arquivos, uma falha em definir índices apropriados pode e geralmente leva a uma série de problemas de acessibilidade, problemas com execução de consulta e balanceamento de carga. O MongoDB oferece uma ampla gama de índices e recursos com ordens de classificação específicas do idioma que oferecem suporte a padrões de acesso complexos a conjuntos de dados. Os índices do MongoDB podem ser criados sob demanda para acomodar padrões de consulta e requisitos de aplicativo em constante mudança em tempo real (MongoDB Features, 2021).

Um aspecto importante quando se trabalha com grandes volumes de dados é o balanceamento de carga, que continua sendo um dos grandes desafios do gerenciamento de banco de dados em grande escala. A distribuição adequada de milhões de solicitações pode levar a uma diferença significativa no desempenho. Por meio de recursos de dimensionamento horizontal, como replicação e fragmentação, o MongoDB oferece suporte ao balanceamento de carga em grande escala, a plataforma pode lidar com várias solicitações simultâneas de leitura e gravação para os mesmos dados com o melhor controle de simultaneidade e protocolos de bloqueio que garantem a consistência dos dados. Não há necessidade de adicionar um balanceador de carga externo. O MongoDB garante que cada usuário tenha uma visualização consistente e uma experiência de qualidade com os dados que precisam acessar (MongoDB Features, 2021).

De acordo com o trabalho de (DAMODARAN, 2016), é notado que quando o número de registros inseridos ou pesquisados é menor, não há diferença no tempo de execução de cada bancos de dados MongoDB e MySQL. No entanto, quando o número de registros é aumentado, o MongoDB mostra uma redução significativa no tempo de execução em comparação com o MySQL. Portanto, quando o número de registros é maior, o MongoDB leva menos tempo em comparação com o MySQL. O MongoDB pode ser preferido para melhor desempenho.

Características principais do MongoDB (MongoDB Features, 2021):

1. Alto desempenho: o MongoDB fornece persistência de dados de alto desempenho. Em particular o suporte para modelos de dados, e suporte para queries mais rápidas.

2. Rich Query Language: o MongoDB oferece suporte a uma linguagem de consulta rica para dar suporte a operações de leitura e gravação (CRUD). A facilidade de replicação do MongoDB, chamada conjunto de réplicas, fornece failover automático e redundância de dados.
3. Escalabilidade Horizontal: o MongoDB fornece escalabilidade horizontal como parte de sua funcionalidade principal; Sharding distribui dados por um cluster de máquinas, e a partir da versão 3.4, o MongoDB oferece suporte à criação de zonas de dados com base na chave de shard.
4. Suporte para vários mecanismos de armazenamento: o MongoDB oferece suporte a vários mecanismos de armazenamento, WiredTiger Storage Engine (incluindo suporte para criptografia em repouso) e In-Memory Storage Engine.

5- DESENVOLVIMENTO

O desenvolvimento desse trabalho ocorreu em conjunto com (NETO, 2022) e assim, validar o conceito da proposta. Foram desenvolvidos três programas com o *framework* Moleculer e a arquitetura de microsserviços. Os programas utilizam a ferramenta Docker para executar programas em ambientes Windows, Linux ou MacOS. O Docker é uma ferramenta de código aberto para executar programas em *containers* (Docker, 2022), isolando os programas em serviços e interligados pelo *Transporter*. O *Transporter* é baseado no software RabbitMQ, que interliga os serviços utilizando as suas filas de mensagens. Os programas foram divididos em três partes principais:

OPC UA Orchestrator: é responsável pela organização da produção fabril, possibilita a orientação da produção de itens com base nas rotas do pedido previamente gerenciadas pelo *Order Orchestrator*. Permite o fluxo de comandos de controle para garantir o sucesso da produção e comunicação entre as camadas adjacentes com a da informação;

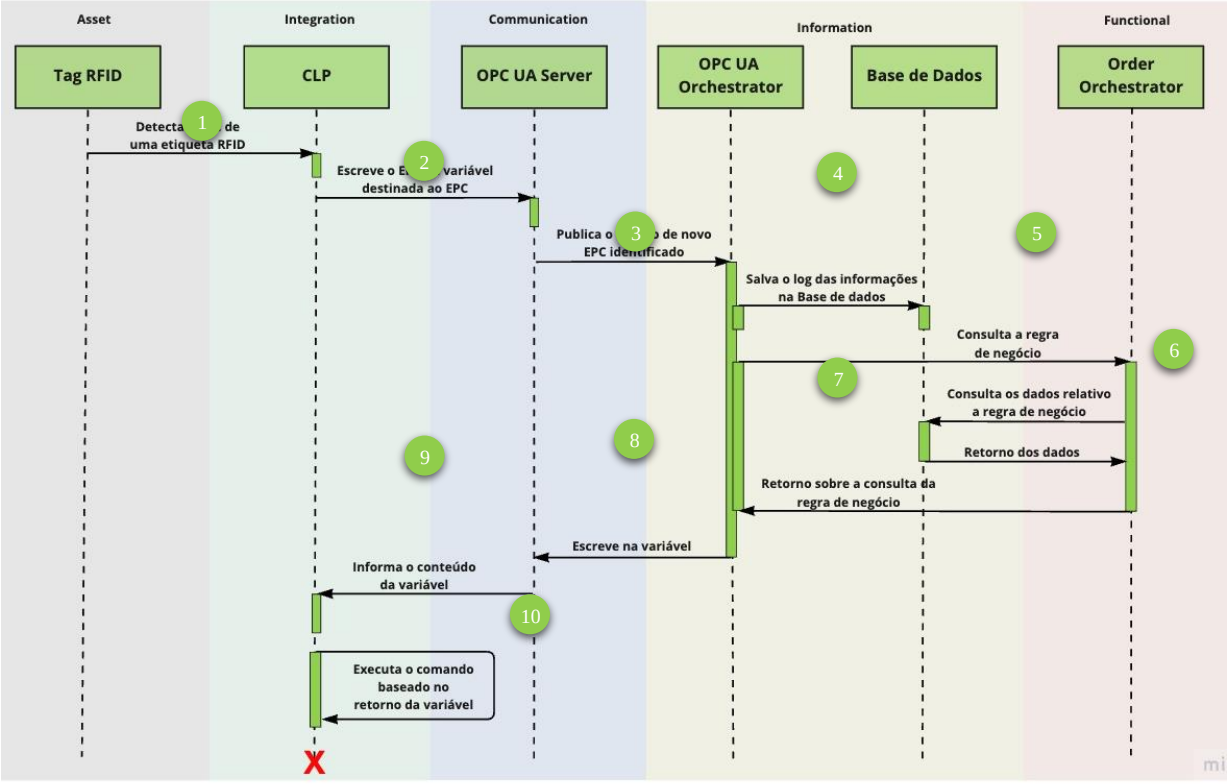
Order Orchestrator: é responsável por organizar os pedidos da produção, definindo os dados e como serão armazenados no banco de dados. Todas as operações de controle entre pedidos de itens na manufatura são tratadas nesse software, incluindo como os pedidos podem mudar e fluir na produção quando o cliente solicita. É importante ressaltar que esta parte está relacionada ao trabalho de (NETO, 2022) e não faz parte dessa pesquisa, junto com o próximo item *API Gateway*;

API Gateway: é a interface responsável por disponibilizar o contrato da API, implementado com a tecnologia API RESTful, para permitir que os clientes se conectem com facilidade e segurança na fábrica. A API permite que as regras de negócio estejam disponíveis para a camada de Negócio do RAMI 4.0. As principais regras de controle do usuário são: criar, visualizar, alterar e excluir pedidos e itens.

A apresentação do fluxo dos dados, responsabilidades e a metodologia de cada camada é muito importante para essa seção. A Figura 17 desenha o diagrama de sequência com as principais operações realizadas sobre a implementação dos programas citados: OPC UA Orchestrator e Order Orchestrator. Esse diagrama relaciona as camadas de Ativos (*Asset*) até a Funcional (*Functional*) do RAMI 4.0, com o fluxo de dados geral. O fluxo de dados pode demandar operações adicionais de tráfego de dados entre os programas, mas a modelagem geral foi extraída com base

empírica das principais operações implementadas nesse trabalho.

Figura 17 - Fluxo geral da informação entre as camadas do RAMI 4.0.



Fonte: Autoria própria.

A camada Funcional (*Functional*) é a responsável por endereçar e fornecer as regras de negócios para cada EPC lido proveniente da camada de Ativos (*Asset*), do ponto de vista da fábrica. Além disso, ele lida com todas as informações da perspectiva do cliente.

A camada da Informação (*Information*) possui a modelagem da Informação no banco de dados e o OPC UA *Orquestrador* é responsável por toda a manipulação de dados entre as camadas adjacentes como princípio. Ou seja, sem o OPC UA Orquestrador, o OPC UA *Server* não seria capaz de realizar a conexão direta no banco de dados, salvando seus dados de acordo com a modelagem proposta e controle das regras de negócio.

Na Figura 17, o fluxo se inicia com uma leitura pela antena RFID, representando uma entrada no sistema de um EPC. A camada de Ativos (*Asset*) é a origem da leitura e fornece o dado para o CLP, que possui o servidor OPC UA como parte principal de software de comunicação e as variáveis configuradas para que o EPC possa ser inserido após a leitura. Assim, a leitura do dado EPC se torna disponível para as

interações das camadas superiores até a Funcional (*Functional*) do RAMI 4.0 mapeadas nesse trabalho.

As setas da Figura 17 representam o fluxo dos dados entre os programas OPC UA Server, OPC UA Orchestrator, Banco de dados e o *Order Orchestrator*, que se inicia com o evento de leitura até o retorno da informação processada nas camadas superiores para o CLP. As numerações da Figura 17 corresponde aos números e descrições:

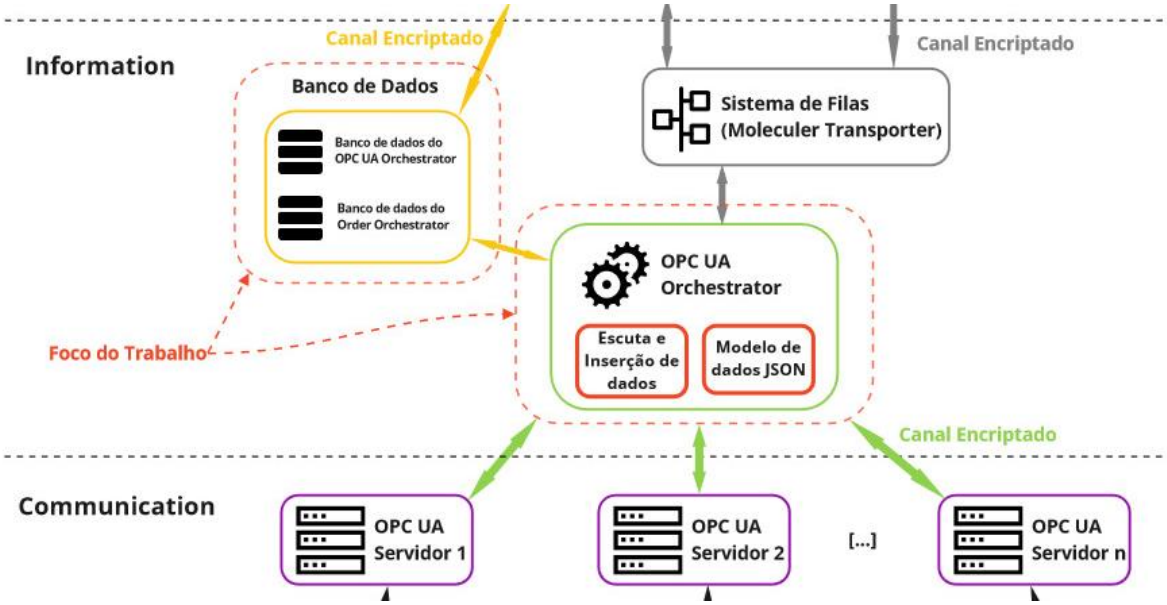
1. A antena realiza a leitura do dado EPC gravado na *tag* RFID;
2. O hardware do CLP, que é apoiado pelo leitor RFID, identifica o valor do EPC, mantem em memória temporária e envia logo após para o OPC UA *Server*;
3. O OPC UA *Server* armazena o valor da leitura do EPC mais atual e dispara um evento de novo EPC identificado no raio da antena RFID, na estação da manufatura;
4. O OPC UA *Orchestrator* recebe o evento com o novo valor do EPC e salva no banco de dados. Nesse momento, OPC UA *Orchestrator* possui a identificação do número da estação que realizou a leitura do EPC, baseado no mapeamento prévio do endereço IP da estação na rede local;
5. O OPC UA *Orchestrator* inicia a consulta da regra de negócio associada a leitura do novo valor do EPC na estação e aguarda o retorno do *Order Orchestrator*;
6. O *Order Orchestrator* consulta o banco de dados para obter os dados de rastreamento do produto na produção, baseado no valor do EPC e identificação da estação. Assim, a regra de negócio é aplicada baseada nas informações de retorno do banco de dados;
7. O *Order Orchestrator* retorna à informação processada para o OPC UA *Orchestrator*;
8. O OPC UA *Orchestrator* escreve o dado necessário na variável de retorno do OPC UA *Server*, que apoia a tomada de decisão nas camadas inferiores;
9. O OPC UA *Server* informa o valor da variável de retorno para o CLP por meio de um evento;
10. O CLP executa a ação baseado no retorno. Nesse ponto, o CLP pode executar diversas ações, como ativar um sensor de luz ou acionar um motor de passo, por exemplo.

As subseções a seguir tem como objetivo apresentar detalhes dos resultados obtidos sobre a implementação das camadas de Comunicação e Informação, descrição da prova de conceito e, por fim, as discussões e considerações finais.

5.1. Camada de Comunicação

A camada de Comunicação é responsável pela interação entre os itens do chão de fábrica e orquestradores de dados (servidor OPC UA), responsável por guiar as informações de maneira clara e estruturada. O protocolo de comunicação utilizado foi o OPC UA. A configuração do servidor OPC UA utiliza o Node-RED e é o componente principal de software que implementa o protocolo OPC UA nessa camada. Na Figura 18, pode-se observar a distribuição dos servidores OPC UA na camada de Comunicação (*Communication*).

Figura 18 - Arquitetura das Camadas de Comunicação e Informação.



Fonte: Autoria própria.

Estes entes são responsáveis por todos os dados que passam pela fábrica, tantos os estímulos das estações RFID, quanto as respostas do sistema sobre rotas de produção nesse trabalho. O OPC UA *Server* é responsável por saber de onde todas as informações são providas e para onde elas têm que ir por meio de eventos, que sinalizam novas leituras ou escritas de um dado, assim como, aguardar os respectivos retornos em variáveis destinadas a isso.

5.2. Camada da Informação

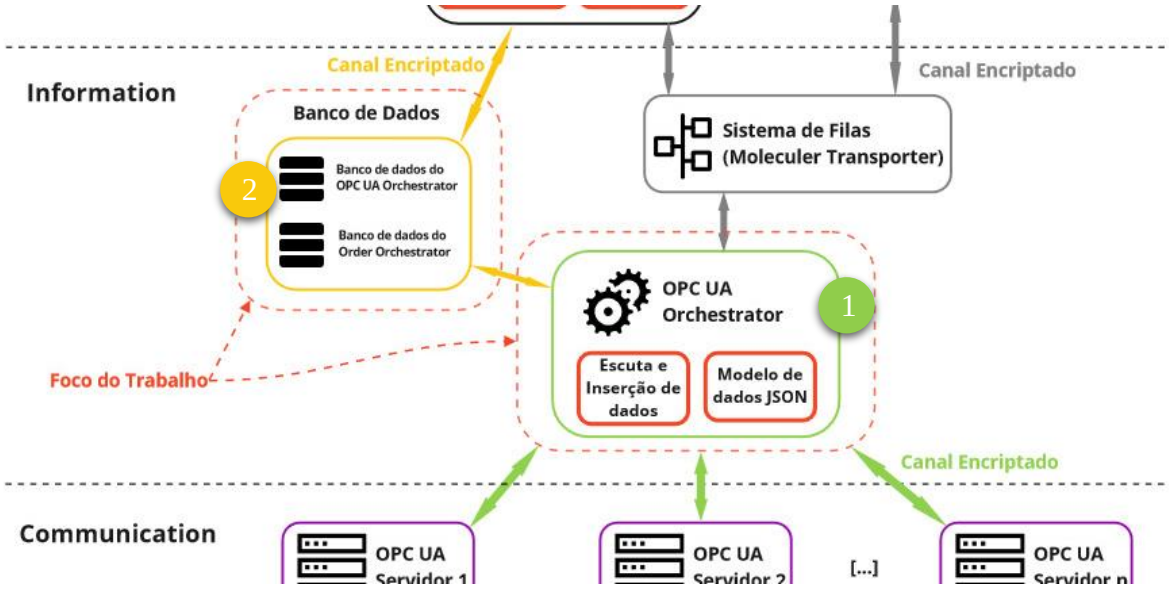
Na Figura 18, pode ser visto a arquitetura da camada da Informação e nela temos dois destaques principais, o OPC UA *Orchestrator* e o Banco de Dados com a modelagem da informação. O OPC UA *Orchestrator* é construído na linguagem de programação JavaScript e é executado no Node.js. O OPC UA *Orchestrator* possui também o cliente do protocolo de comunicação OPC UA. Além do JavaScript e do Node.js, é usado o *framework* Moleculer para a recepção agnóstica de informações que fluem neste ambiente e define a arquitetura em microsserviços.

Enquanto o OPC UA *Server* identifica aonde cada informação vem e para onde ela vai, o OPC UA *Orchestrator* sabe o que fazer com ela, seja consultar regras de negócio, inserir no banco dados de controles, geralmente de sensores, ou chamar outras rotinas e serviços. Toda a comunicação é feita através do *framework* Moleculer, que abre um caminho direto com o *Order Orchestrator* na camada Funcional, questionando sobre regras de negócio, pedidos para inserir, remover e alterar dados relativos as regras de negócio. O desenvolvimento da Camada Funcional faz parte de outro trabalho de mestrado (NETO, 2022). Sobre a segurança do tráfego de dados, o Moleculer fornece protocolos de segurança definidos para comunicação entre microsserviços, utilizando o *Transporter*, e tem sua criptografia própria, impedindo vazamentos de dados no meio de comunicação e controle de acesso.

Foram utilizadas as ferramentas discutidas no capítulo 4- Materiais e Métodos para o desenvolvimento. Essas ferramentas formam a base do projeto e apoiam a codificação base do OPC UA *Orchestrator*, com o objetivo de validar a viabilidade técnica da implementação. As principais partes básicas onde ocorre o fluxo de dados são observadas na Figura 19, onde dois componentes cruciais e já citados nessa seção estão assinalados com os números correspondentes. A lista abaixo é avaliada ao longo desta seção em subseções com maiores detalhes:

1. O OPC UA *Orchestrator*;
2. A modelagem da informação no banco de dados.

Figura 19 - Partes principais da camada da informação.



Fonte: Autoria Própria.

O OPC UA *Orchestrator*, o *Order Orchestrator* e a API *Gateway*, desenvolvido para esse trabalho, formam a principal parte da prova de conceito da proposta. O OPC UA *Server* foi criado com o Node-RED para simular a orquestração nas camadas inferiores do RAMI 4.0 e tornar possível a demonstração da interação com a fábrica.

Nessa seção, é abordado o desenvolvimento pertinente ao escopo desse trabalho, focado na camada da Informação, com a visão do banco de dados, modelagem da Informação e o programa OPC UA *Orchestrator*.

5.2.1. OPC UA Orchestrator

O protocolo preferencial definido nas especificações do RAMI 4.0 é o OPC UA. Assim, a escolha do protocolo na camada de Informação foi partindo do princípio de que o todos os elementos servidores de dados presentes são servidores OPC UA na camada da Comunicação. Portanto, é necessário um elemento presente na camada de Informação que seja capaz de coletar ou gerenciar os dados que são produzidos pelos servidores OPC UA. Esse elemento foi definido com o nome de OPC UA *Orchestrator* e é um programa escrito em JavaScript com o *framework* Moleculer.

O OPC UA *Orchestrator* possui um serviço específico que incorpora a biblioteca node-opcua para implementar o cliente de acesso ao OPC UA *Server*. Cada serviço cliente tem a responsabilidade de conectar em um OPC UA *Server* diferente na

manufatura e gerenciar a troca das informações entre o chão de fábrica e as camadas superiores do RAMI 4.0, com seus componentes de software. Os recursos de hardware e software distribuídos pela indústria no chão de fábrica coleta os dados das camadas inferiores, representadas pelas estações de trabalho, que coletam dados de hardwares como: CLPs, leitores RFIDs e sensores. Os dados são coletados e transportados pelo OPC UA *Orchestrator* para as camadas superiores e faz a correspondência dos dados seguindo a modelagem da informação no banco de dados.

O OPC UA *Orchestrator* é implementado seguindo a arquitetura de microsserviços do Moleculer, que possibilita a conexão com outros microsserviços via um sistema de filas (Moleculer *Transporter*). O *Order Orchestrator* é acessado por meio das ligações de filas pelo OPC UA *Orchestrator* e vice-versa.

O Moleculer oferece bibliotecas padrões ou clientes para conexão com diversos tipos de bancos de dados diferentes. Assim, o OPC UA *Orchestrator* utilizou uma biblioteca padrão já incorporada para acessar o banco de dados MongoDB, onde os dados serão persistidos ou acessados para a tomada de decisões na manufatura. A transformação do padrão da informação industrial em um padrão que se pode trafegar na arquitetura de microsserviços, habilita a solução para o uso nos sistemas de TI e torna o sistema escalável, que é uma das principais características de uso de uma arquitetura de microsserviços. Outras características que compõe uma arquitetura de microsserviços são:

- suporta a arquitetura orientada a eventos;
- eventos com balanceamento de carga para distribuição das tarefas entre os serviços;
- recursos de tolerância a falhas, com a replicação de várias instâncias, evitando indisponibilidade do microsserviço;
- implementação de cada serviço com código especializado e separação clara de contexto da aplicação;
- acesso isolado de cada serviço em relação a sua própria base de dados.

5.2.2. Modelagem da Informação

No RAMI 4.0, a camada da Informação necessita ter o modelo da informação definido e este deve contemplar a modelagem da fábrica e seus ativos (*assets*),

relacionados à produção e ciclo de vida do produto. A estrutura da informação tem que ser escalável de forma a incorporar novos elementos, especializada com a aplicação a ser desenvolvida para a manufatura e controle de processos de produção.

O trabalho de modelagem da informação é complexo quando utilizamos várias fontes de dados diferentes na manufatura. Então, foram observadas formas de simplificar e direcionar os dados utilizando modelos de dados já existentes na literatura relacionada. Baseado no estudo do AutomationML e o OCG SensorThings API realizado na revisão bibliográfica, foi selecionado o OCG SensorThings API como base nesse processo.

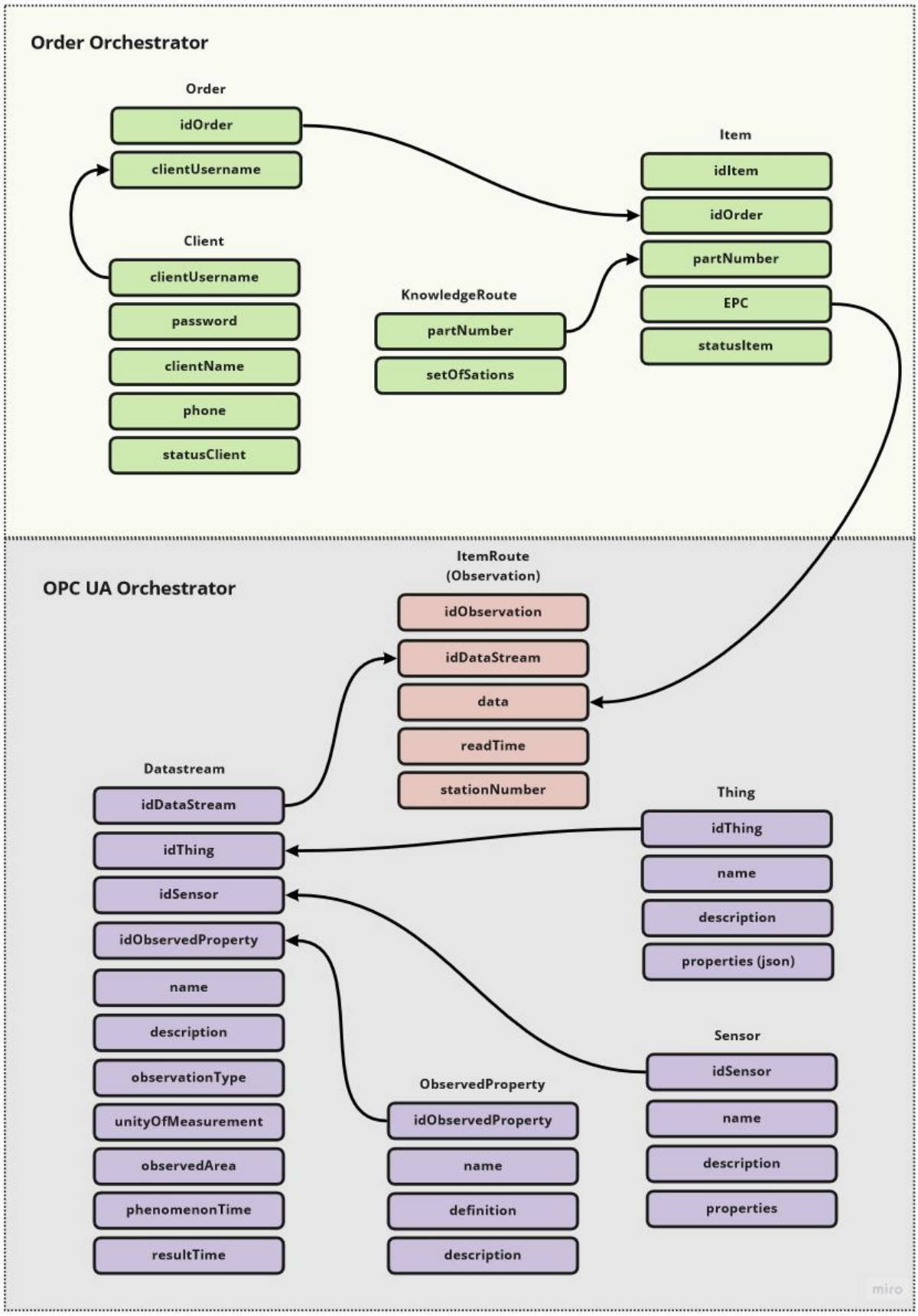
A estrutura oferecida pelo OCG SensorThings API é ampla e a possibilidade em aplicar simplificações é facilitada por utilizar uma estrutura UML. Assim, para esse trabalho que contempla a manufatura flexível, a simplificação aplicada manteve a coerência dos dados com flexibilidade para mudanças e aplicações de diversos cenários de mapeamento de objetos físicos para o virtual.

A estrutura UML do OCG SensorThings API possui um modelo da informação de fácil transição para outros modelos como o JSON, que é focado nesse trabalho. O formato JSON possibilita:

- Simplicidade e menor curva de aprendizado na modelagem da informação;
- Flexibilidade na construção de documentos JSON com a possibilidade da variação da estrutura de dados;
- Um maior desempenho dos componentes de TI diminuindo a necessidade da transformação de dados entre sistemas utilizando o formato JSON. O banco MongoDB utiliza o formado BSON (JSON binário) e o framework Moleculer utiliza objetos JavaScript que são objetos totalmente intercambiáveis e próximos do formato JSON;
- A construção de API RESTful disponibilizada na API *Gateway* da camada Funcional, por utilizar o documento JSON para as respostas direcionadas a camada de Negócio.

A modelagem da informação no banco de dados é contemplada na camada da Informação. Na Figura 20, pode-se ver a estrutura das *Collections*, que nada mais são do que as tabelas no padrão do banco de dados MongoDB. A camada da informação é a principal responsável por guardar os dados utilizados ou provenientes de outras camadas do modelo RAMI 4.0.

Figura 20 - Modelagem Geral do Banco de Dados.



Fonte: Autoria Própria.

O banco de dados MongoDB possibilita o relacionamento entre tabelas ou *collections* garantidos por meio do código da aplicação e utilizando a propriedade de identificação de cada registro. Assim, mantendo a consistência dos dados e a velocidade característica de um banco de dados NoSQL como o MongoDB.

As *Collections* Thing, Datastream, ItemRoute, ObservedProperty e Sensor são implementas no padrão proposto pelo SensorThings API, pertencente ao Open Geospatial Consortium (OGC), e dizem respeito a modelagem dos dados pertencentes a fábrica e objetos físicos presentes nela. Os dados são provenientes das camadas de Asset, Integration e Communication no modelo RAMI 4.0.

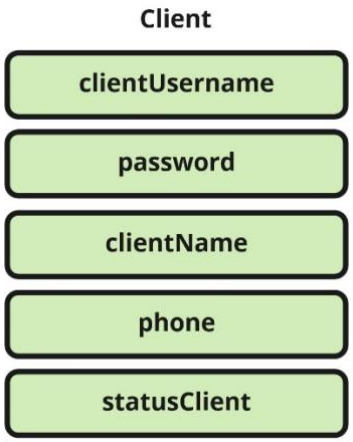
Os dados provenientes das camadas adjacentes superiores (Functional e Business), refletem os dados do cliente na *Collection Client*, dados do pedido nas *Collection Order*, com as informações mais específicas da Ordem na *Collection Item*, e pôr fim, a base de conhecimento, ou seja, as regras de fabricação de qualquer item na *Collection* KnowledgeRoute.

A maioria das *collection* nessa modelagem possuem os campos ou propriedades *createdAt* e *updatedAt*. A identificação temporal de quando foi criado o registro é inserido no campo *createdAt* e a última data quando é realizado uma modificação no campo *updatedAt*. O valor *null* é comum no campo *updatedAt* e significa que não foi realizado qualquer alteração no registro da *collection* alvo. A falta ou não dos dois campos não implica a invalidade da solução.

As seguintes tabelas, também chamadas de coleções ou *collection* em inglês, foram modeladas para banco de dados NoSQL e listadas a seguir com as principais propriedades:

Collection Client define os dados do cliente dentro da fábrica, como nome de usuário, senha, e-mail, telefone e um *status* que permite ou não a produção. A estrutura é representada na Figura 21.

Figura 21 - Collection Client.



Fonte: Autoria Própria.

- clientUsername é usado para identificar o cliente;
- password é usado para armazenamento de senhas dos clientes;
- clientName armazena o nome dos clientes;
- email armazena o e-mail de clientes;
- phone é usado para armazenar o telefone dos clientes;
- statusClient tem os valores de “inactive” se o cliente estiver suspenso e “active” se o cliente estiver ativo. Somente clientes ativos podem solicitar novos pedidos.

O documento JSON da Figura 22 é utilizado nesse trabalho e diz respeito a *collection Client*.

Figura 22 - Exemplo do documento JSON Collection Client.

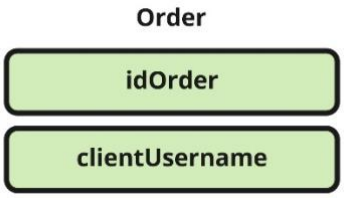
Client


```
{
   "clientUsername" : "joel.ravelli@gmail.com",
   "password" : "pass123)[45",
   "clientName" : "Joel R. J.",
   "phone" : "5515222227777",
   "statusClient" : "active",
   "createdAt" : "2021-08-16T02:40:10.453Z",
   "updatedAt" : "2022-04-08T10:10:21.120Z"
}
```


Fonte: Autoria Própria.

Collection Order é onde são armazenados os pedidos relativos as ordens dos clientes, com o id da ordem e o nome do cliente dentro da fábrica. A estrutura é representada na Figura 23.

Figura 23 - Collection Order.

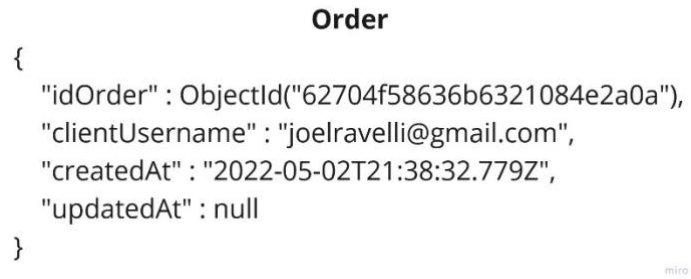


Fonte: Autoria Própria.

- idOrder é um identificador incremental, chave primária e relaciona um determinado pedido de um cliente específico;
- clientUsername é a chave primária, utilizada para identificar o cliente, uma vez que somente clientes cadastrados podem solicitar pedidos.

O documento JSON da Figura 24 é utilizado nesse trabalho e diz respeito a *collection* Order.

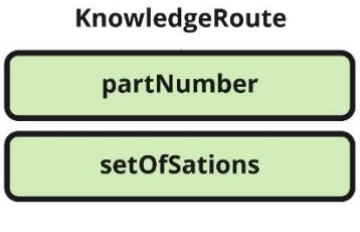
Figura 24 - Exemplo do documento JSON Collection Order.



Fonte: Autoria Própria.

Collection KnowledgeRoute é onde são armazenadas as regras de conhecimento, ou regra de produção da fábrica. As informações que os sistemas de manufatura flexível vão buscar para saber qual a rota e o passo-a-passo da produção dentro da fábrica. Seguindo uma rota pertinente ao PartNumber ou tipo de produto. A estrutura é representada na Figura 25.

Figura 25 - Collection KnowledgeRoute.



Fonte: Autoria Própria.

- partNumber é a chave primária, usada para especificar o tipo de item;
- setOfStations fornece a lista de estações que este PN específico deve navegar para se tornar um bem final dentro da fábrica.

O documento JSON da Figura 26 é utilizado nesse trabalho e diz respeito a *collection KnowledgeRoute*.

Figura 26 - Exemplo do documento JSON Collection KnowledgeRoute.

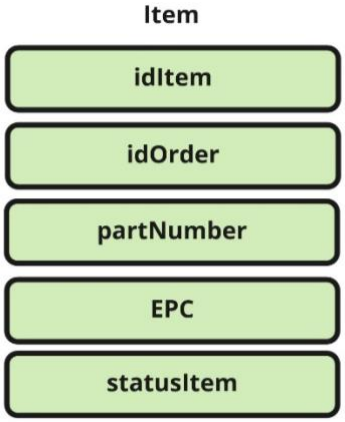
KnowledgeRoute


```
{
  "partNumber" : "AAAA",
  "setOfStations" : [
    1,
    5
  ],
  "createdAt" : "2022-04-20T02:07:31.056Z",
  "updatedAt" : null
}
```


Fonte: Autoria Própria.

Collection Item define todos os itens que serão produzidos dentro da ordem e PartNumber ou produto requisitado. Cada item é individualizado para garantir controle e sempre possui um número de Ordem e PartNumber associado a ele, para identificar o bem final a ser produzido. Todo Item tem um identificador único e um EPC a ser gravado dentro de sua *tag* RFID, para identificar a qualquer momento o produto dentro da fábrica. A estrutura é representada na Figura 27.

Figura 27 - Collection Item.



Fonte: Autoria Própria.

- idItem é incremental, chave primária, usada como identificador único de um item;
- idOrder é um identificador incremental, chave primária, usada para correlacionar um item a um determinado pedido;
- partNumber, é incremental, chave primária, usada para identificar o item pelo número da peça;
- EPC é número serial com valor único armazenado na *tag* RFID que é fixada no produto para identificação;
- statusItem pode ser “not_started” se o pedido ainda não começou a ser produzido, “in_production” se já foi iniciado, “finished” se for concluído.

O documento JSON da Figura 28 é utilizado nesse trabalho e diz respeito a *collection Item*.

Figura 28 - Exemplo do documento JSON Collection Item.

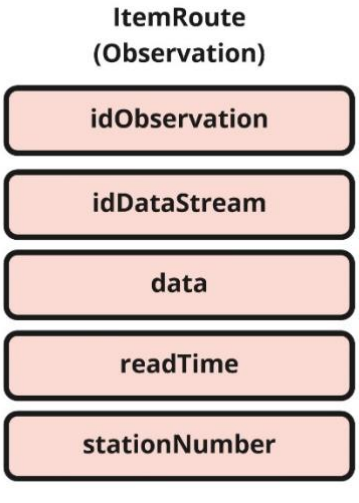
Item


```
{
   "idItem" : ObjectId("62704f58636b6321084e2a0b"),
   "idOrder" : ObjectId("62704f58636b6321084e2a0a"),
   "partNumber" : "AAAA",
   "EPC" : "000000000028",
   "statusItem" : "not_started",
   "createdAt" : "2022-05-02T21:38:32.797Z",
   "updatedAt" : null
}
```


Fonte: Autoria Própria.

Collection ItemRoute é uma visão temporal dos registros de tudo o que é lido na fábrica pelas estações relacionados ao rastreamento de produtos com *tags* RFID, armazenando todos os dados e permitindo que as informações sejam cruzadas posteriormente para a tomada de decisão do sistema. A estrutura é representada na Figura 29.

Figura 29 - Collection ItemRoute.

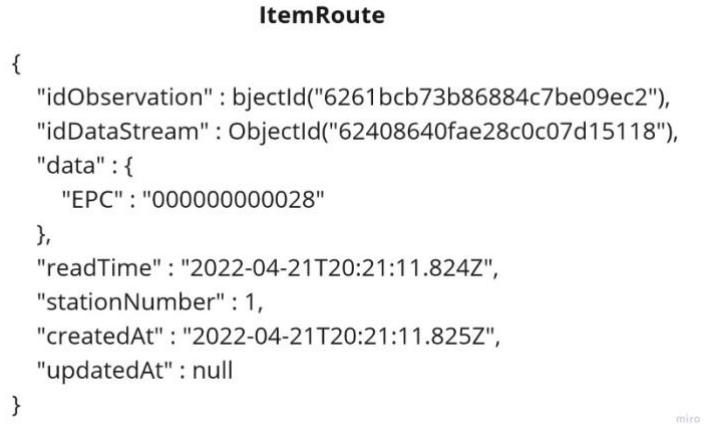


Fonte: Autoria Própria.

- idObservation é um identificador incremental, chave primária, usada como identificador único da observação do item na fábrica;
- idDataStream relaciona com *Collection Datastream*;
- data é usado para armazenar um EPC lido pelo leitor RFID;
- readTime é usado para armazenar quando o item foi lido pelo leitor RFID;
- stationNumber identifica a estação RFID em que o item foi lido.

O documento JSON da Figura 30 é utilizado nesse trabalho e diz respeito a *collection ItemRoute*.

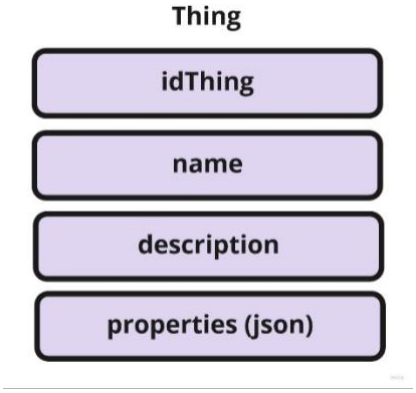
Figura 30 - Exemplo do documento JSON Collection ItemRoute.



Fonte: Autoria Própria.

Collection Thing é onde os dados de identificação da coisa estarão armazenados, ou seja, as informações básicas do objeto físico descrito no banco de dados. A estrutura é representada na Figura 31.

Figura 31 - Collection Thing.



Fonte: Autoria Própria.

- idThing é identificador incremental, chave primária, usada como identificador único do objeto na fábrica;
- name é o nome da “coisa” que queremos armazenar, ou seja, qualquer objeto real que queremos realizar observações na fábrica;
- description é uma descrição mais detalhada sobre o objeto;
- properties contém propriedades disponibilizadas pelo usuário na forma de pares chave-valor como documento interno JSON.

O documento JSON da Figura 32 é utilizado nesse trabalho e diz respeito a *collection Thing*.

Figura 32 - Exemplo do documento JSON Collection Thing.

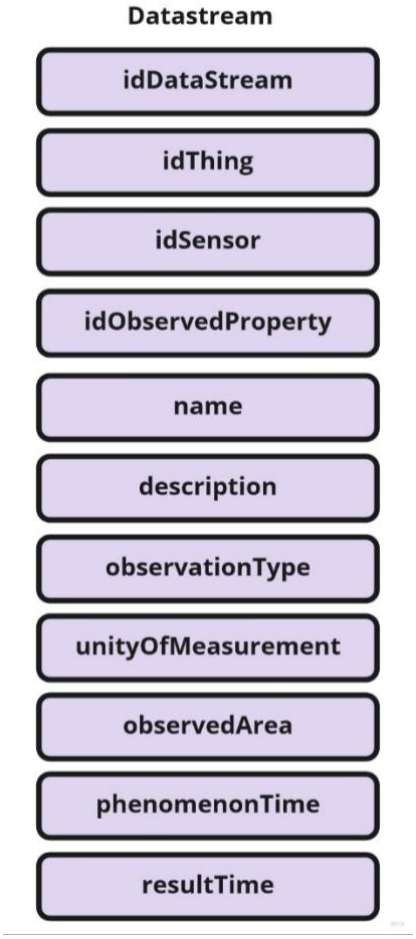
Thing


```
{
  "idThing" : ObjectId("628aa2c6d7b6eabf82cab92d"),
  "name" : "PLC Station",
  "description" : "PLC system to control and monitoring the station functions",
  "properties" : {
    "factoryId" : "32562117",
    "stationHardVersion" : "v1.01",
    "stationSoftVersion" : "v2.05"
  }
}
```


Fonte: Autoria Própria.

Collection Datastream é onde os dados dos sensores e observações produzidas pelos sensores são relacionados com a coisa (*thing*). Essa coleção é fundamental no modelo de dados para manter a relação entre os dados. A estrutura é representada na Figura 33.

Figura 33 - Collection Datastream.

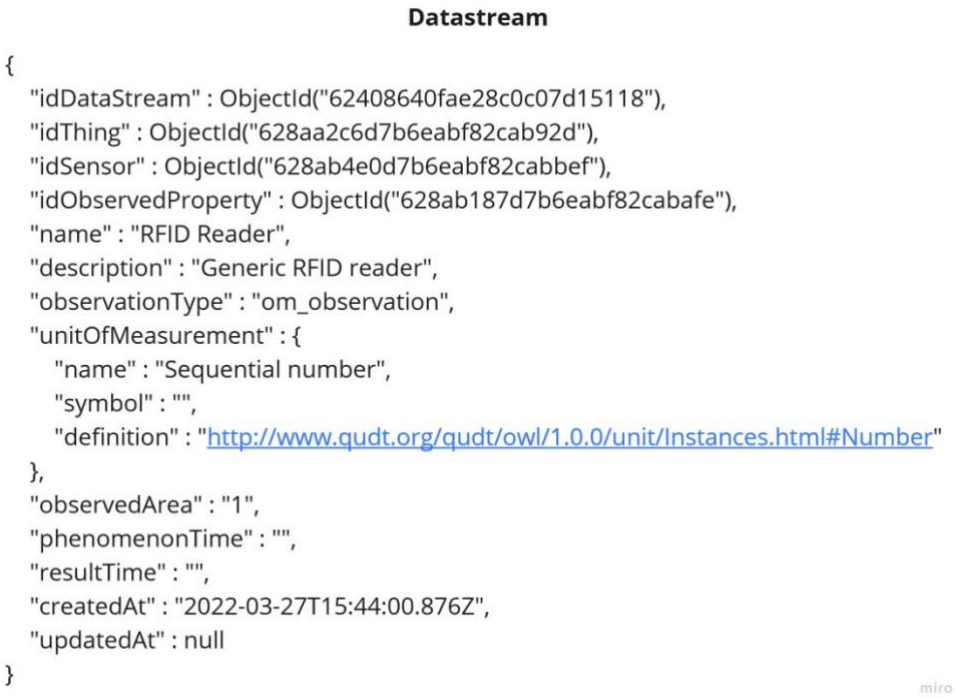


Fonte: Autoria Própria.

- idDataStream é identificador incremental, chave primária, usada como identificador único da observação;
- idThing é o identificador da coisa (*thing*) modelada no banco;
- idSensor é o identificador do sensor associado a observação;
- idObservedProperty é o identificador da propriedade observada;
- name é o nome descritivo para a entidade Datastream, que armazenando as propriedades que são lidas de um sensor;
- description é uma descrição mais detalhada sobre as propriedades;
- observationType relaciona a codificação da unidade de medida;
- unitOfMeasurement é um objeto JSON e possui três pares de chave-valor. A propriedade *name* apresenta o nome da unitOfMeasurement, que diz respeito ao nome da unidade de medida; a propriedade com o nome *symbol* com o símbolo da unidade; e a propriedade *definition*, que contém o URI que define o unitOfMeasurement;
- observedArea é uma descrição espacial das observações do Datastream;
- phenomenonTime é o intervalo de tempo fixo em que ocorre o fenômeno de leituras pertencentes a este Datastream;
- resultTime é o intervalo temporal de produção dos resultados das observações pertencentes a este Datastream.

O documento JSON da Figura 34 é utilizado nesse trabalho e diz respeito a *collection Datastream*.

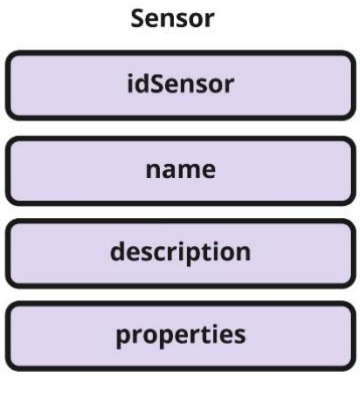
Figura 34 - Exemplo do documento JSON Collection Datastream.



Fonte: Autoria Própria.

Collection Sensor é onde a descrição do sensor físico é salva com o objetivo observar uma propriedade ou um fenômeno. A estrutura é representada na Figura 35.

Figura 35 - Collection Sensor.



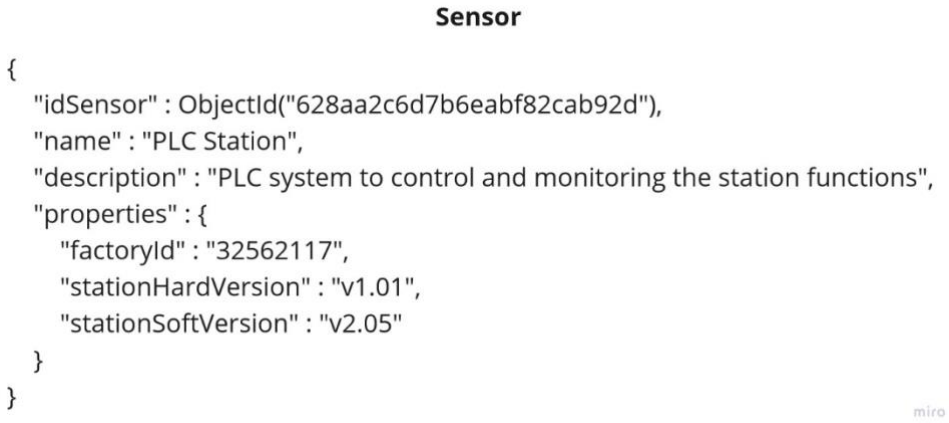
Fonte: Autoria Própria.

- idSensor é o identificador incremental, chave primária, usada como identificador único do sensor;
- name é o nome descritivo do sensor pertencente a entidade Thing;

- description é uma descrição mais detalhada sobre das propriedades;
- properties é a metadata que descreve o sensor ou o sistema.

O documento JSON da Figura 36 é utilizado nesse trabalho e diz respeito a *collection Sensor*.

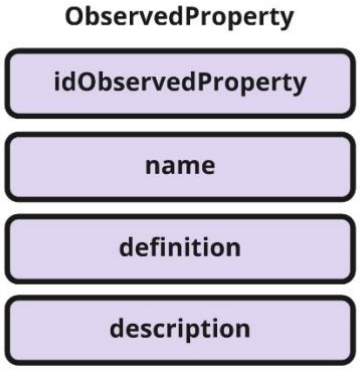
Figura 36 - Exemplo do documento JSON Collection Sensor.



Fonte: Autoria Própria.

Collection ObservedProperty especifica o fenômeno de uma observação provenientes da *Collection ItemRoute* com mais detalhes e informações. A estrutura é representada na Figura 37.

Figura 37 - Collection ObservedProperty.



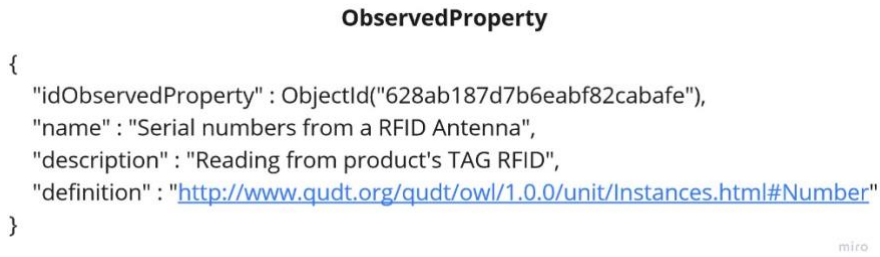
Fonte: Autoria Própria.

- idObservedProperty é o identificador incremental, chave primária, usada como identificador único da propriedade da observação;
- name é o nome descritivo da *ObservedProperty*;
- definition contém o URI que define a propriedade da observação;

- description é uma descrição mais detalhada sobre a *ObservedProperty*.

O documento JSON da Figura 38 é utilizado nesse trabalho e diz respeito a *collection ObservedProperty*.

Figura 38 - Exemplo do documento JSON Collection ObservedProperty.



Fonte: Autoria Própria.

5.3. Prova de Conceito

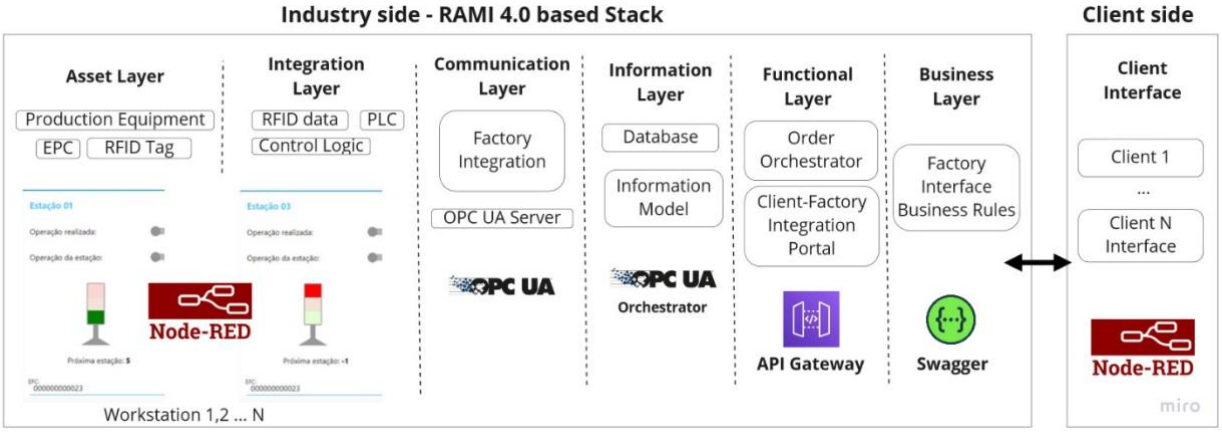
Nesta seção, os resultados experimentais são demonstrados de acordo com a proposta da arquitetura desse trabalho. Os componentes presentes nos cenários de uma fábrica são: clientes, estações de trabalho, pedidos de produtos e produtos em uma manufatura flexível. Optou-se pela apresentação dos resultados da prova de conceito por meio de uma abordagem assíncrona, pois o sistema foi construído na arquitetura de microsserviços baseados em eventos assíncronos. As ferramentas utilizadas para implementação da prova de conceito estão na seção 5.3.1, junto com suas correspondências das camadas do RAMI 4.0. Os cenários definidos serão apresentados na prova de conceito da seção 5.3.2, com os componentes de hardware e software que compõem a simulação da manufatura flexível. Na seção 5.3.3 é executado dois cenários de teste: um de falha e outro do fluxo normal, para estressar o fluxo de produção de um pedido na manufatura.

5.3.1. Ferramentas da Prova de Conceito

Todas as ferramentas e sistemas desenvolvidos nesta prova de conceito, desde a estação de trabalho até a interface do cliente, são mostrados na Figura 39. O lado da indústria representa os elementos de fábrica que são alocados de acordo com as camadas RAMI 4.0 da arquitetura desenvolvida. O lado do cliente representa o mundo externo, no qual o cliente irá interagir com a fábrica para criar e gerenciar seus pedidos na produção. É muito importante reforçar que esta prova de conceito é apenas um exemplo de um sistema de manufatura flexível e que a arquitetura de

microsserviços desenvolvida nesta pesquisa poderia ser aplicada em outros tipos de produção. Existe uma possibilidade ampla de implementações, dependendo do cliente, da linha de produção, das configurações das estações de trabalho, dos produtos e as especificações dos pedidos.

Figura 39 - Prova de conceito: detalhes das tecnologias e componentes utilizados e suas interações entre camadas do RAMI 4.0.



Fonte: Neto (NETO, 2022).

Da camada de ativos ao lado do cliente (*cliente side*), representado na Figura 39, os componentes da arquitetura podem ser resumidos como:

- **Camada de ativos (Asset Layer):** está relacionada aos equipamentos de produção (máquinas para produzir roupas nesta prova de conceito), a etiqueta RFID no item com seu EPC (Código Eletrônico do Produto), que contém o código de especificação do produto (serial). A prova de conceito dessa camada é representada por parte das estações de trabalho, mais especificamente pelas etiquetas RFID, a antena RFID, os transportadores, LED e sensores. As 5 estações de trabalho foram simuladas utilizando a ferramenta Node-RED;
- **Camada de integração (Integration Layer):** está relacionada aos equipamentos de controle (exemplo: CLP), lógica de controle dos equipamentos de produção, os dados RFID e o suporte de comunicação. Na prova de conceito, a camada é representada por parte das estações de trabalho responsáveis pelo código EPC (dados RFID) e a rede de comunicação fabril. Também, os componentes da camada estão inclusos na simulação Node-RED;
- **Camada de comunicação (Communication Layer):** o RAMI 4.0 recomenda o protocolo OPC UA para a rede de comunicação. Essa camada é responsável por vincular os dados da fábrica a camada de Informação (banco de dados),

fornecendo a interface entre a fábrica e a gestão da produção (camada funcional);

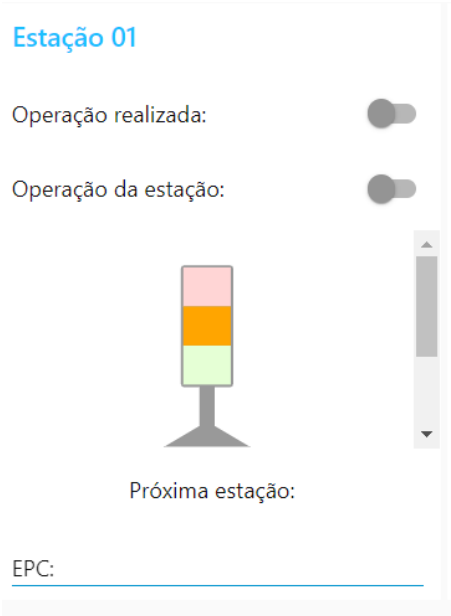
- **Camada de informação (Information Layer):** contém a modelagem da informação e o banco de dados no sistema. É uma parte crucial para a eficiência do sistema e confiabilidade das informações. Nesta prova de conceito, foi escolhido o MongoDB como banco de dados e foi criado o programa OPC UA *Orchestrator*, que funciona como cliente OPC UA e do banco de dados, se conectando ao servidor OPC UA de cada estação de trabalho, para permitir a integração da rede e fluxo de dados da fábrica;
- **Camada Funcional (Functional Layer):** é composta pelo orquestrador de pedidos (Order Orchestrator), que interage com os dois lados (cliente e fábrica), para a tomada de decisões sobre a gestão da produção dos pedidos realizados pelos clientes. Isso ocorre pela disponibilização de uma API de integração para o cliente, permitindo acessar as informações da fábrica por meio da Internet, recebendo e enviando informações de forma segura e com canal encriptado de comunicação. O programa foi criado utilizando a linguagem de programação JavaScript;
- **Camada de negócios (Business Layer):** é composta pelas regras de negócios da fábrica para orientar os clientes sobre como se integrar com a fábrica, fazer pedidos e gerenciá-los. Todo o conhecimento foi reunido e fornecido por uma documentação de APIs (Swagger);
- **Interface do Cliente (Client Side):** mostra um exemplo de como um cliente pode construir uma interface usando a documentação do Swagger e integrando com a API para realizar pedidos, pesquisar, atualizar e deletar pedidos. A prova de conceito apresenta apenas uma implementação do cliente, mas a arquitetura suportaria vários outros clientes. A interface foi criada utilizando a ferramenta Node-RED.

5.3.2. Descrição dos Cenários e Simulação

A comunicação da fábrica com a camada da Informação foi simulada por meio do software Node-RED. As estações de trabalho são responsáveis pelas ações na manufatura e para a prova de conceito, foram criadas cinco estações de trabalho. Cada estação é equipada com um leitor RFID, semáforo e display para interagir com o operador e mostrando a próxima estação. A simulação da estação no *dashboard* do

Node-RED pode ser vista na Figura 40, representando uma estação real e possível de se modelar na fábrica. Para tal foi colocado um botão com a função de operação realizada, com o rótulo “Operação realizada”, que é um *reset* para o leitor RFID, para que não produza outras leituras e zere os campos retornando ao estado inicial da estação. O interruptor “Operação da estação” é utilizado para sinalizar o início da execução na estação quando ativado ou o fim do processo da estação quando desativado. O semáforo é usado para demonstrar se o item deve ou não continuar na sua rota de produção: ele fica verde caso o item esteja na estação certa e está liberado para seguir; laranja se não houver nenhuma interação com o item ou estado inicial; vermelho, caso seja sua última estação ou haja um erro. O campo de visualização da “Próxima estação” retorna a próxima estação que o item deve seguir, 0 se for sua estação final, -1 se for um erro de rota, mas o item existe na base e -2 caso o EPC não exista no base de dados. E por último, existe o campo EPC, onde inserimos manualmente o valor de um EPC para simular a leitura do RFID. Nesse trabalho, não estamos seguindo qualquer padrão de codificação e decodificação de EPC. Porém, isso pode ser realizado adicionando um controle em toda a arquitetura de software e algoritmos como da associação GS1. O EPC é definido como um número sequencial positivo e único para melhor entendimento do trabalho e cenários.

Figura 40 - Demonstração de uma estação de produção.



Fonte: Autoria Própria.

Para a prova de conceito deste trabalho, foi escolhido um cenário de teste que permita simular a flexibilidade do trabalho. Não significa que o trabalho seja limitado a

esse exemplo, pelo contrário, o sistema baseado em microsserviços é passível de fácil adequação para diversos outros cenários e essa foi apenas uma abstração escolhida para demonstração dos resultados. Assim explicado, foi escolhido a padronização a seguir das estações de trabalho, PN e respectivas rotas de conhecimento:

Ações executadas em cada estação:

- Estação 1: Faz o corte no tecido e costura para torná-lo uma camisa;
- Estação 2: Faz o corte no tecido e costura para torná-lo uma camiseta;
- Estação 3: Faz o corte no tecido e costura para torná-lo uma regata;
- Estação 4: Faz o tingimento do tecido na cor preto;
- Estação 5: Faz o tingimento do tecido na cor vermelho;

Itens fabricados de acordo com o PN e estações necessárias para finalizar a produção, possuem as seguintes rotas na manufatura:

- PN: AAAA, produz: camisa vermelha, passa por estações 1 e 5;
- PN: BBBB, produz: camisa preta, passa por estações 1 e 4;
- PN: CCCC, produz: camisa xadrez vermelha e preta, passa por estações 1, 4 e 5;
- PN: DDDD, produz: camiseta vermelha, passa por estações 2 e 5;
- PN: EEEE, produz: camiseta regata preta, passa por estações 3 e 4;

Com as definições acima, usando um PN AAAA e com as informações da rota das estações de trabalho sendo 1 e 5, podemos demonstrar os possíveis resultados na Figura 41. Observe que o EPC correto é o finalizado com o número 23 e o finalizado em 30 não existe no banco de dados, ou seja:

- Na Figura 41, a “Estação 1” com entrada do EPC final 23, se o item estiver na rota certa e não for a última estação: retorna a próxima estação e fica verde, como foi o caso;
- Se o item não existe: retorna -2 e o semáforo fica vermelho. Na Figura 41, a “Estação 02” com entrada do EPC final 30, que não existe no banco de dados;
- Se o item existe no banco de dados, mas está na estação incorreta: retorna -1, e o semáforo fica vermelho. Na Figura 41, a “Estação 03” com entrada do EPC final 23;

- Se a estação está aguardando a passagem do item: o semáforo fica laranja. Na Figura 41, a “Estação 04” está aguardando um item a ser processado;
- Se o item estiver na rota correta e é sua última estação: retorna 0 como próxima estação e fica vermelho. Na Figura 41, a Estação 05 alerta que a peça está pronta com entrada do EPC final 23;

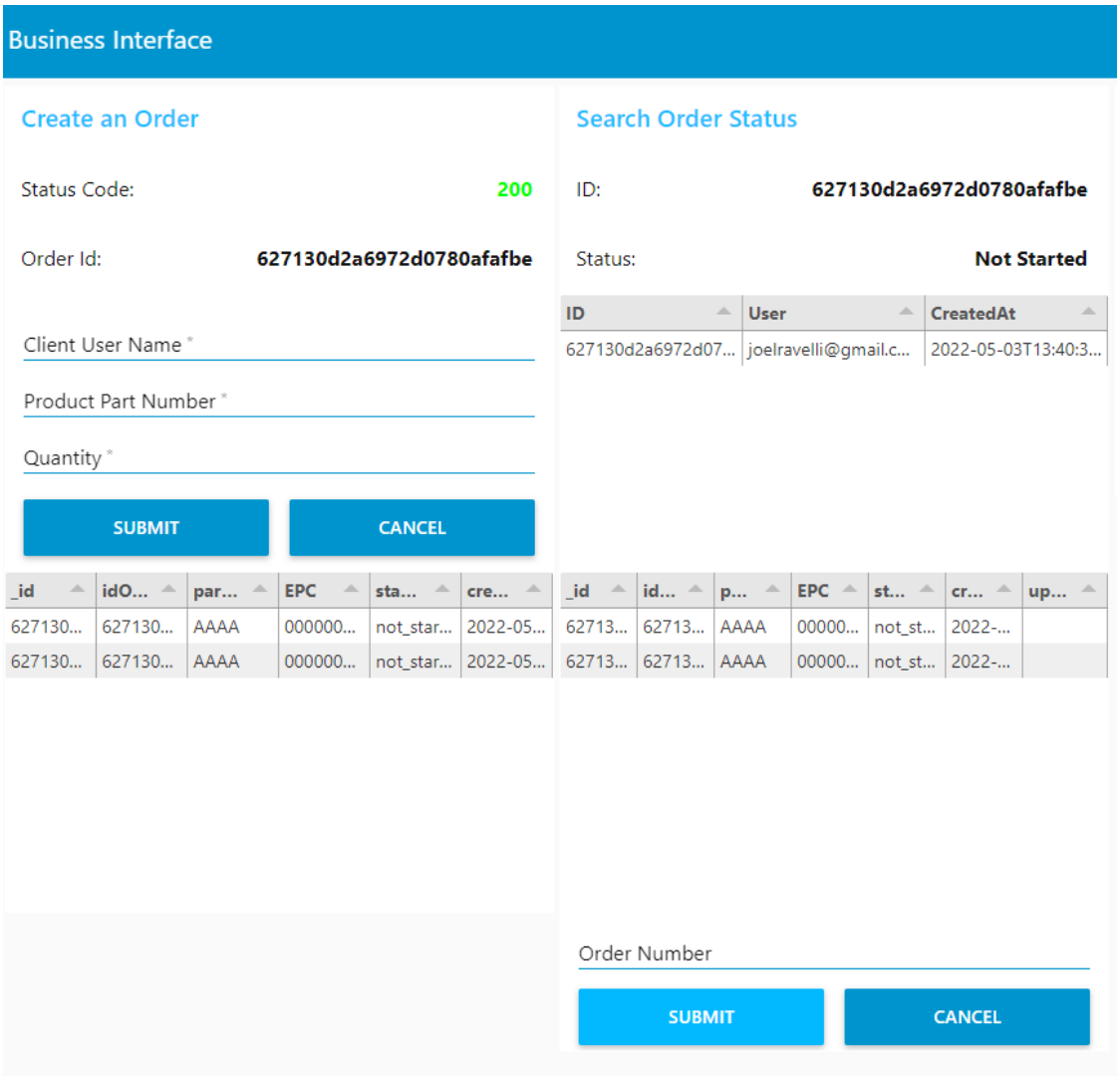
Figura 41 - Estações demostrando os retornos possíveis.

Fonte: Autoria Própria.

5.3.3. Execução de Cenários de Teste

A partir da criação de itens pela camada de negócios, é possível estressar alguns cenários de sucesso e erros. Para isso, foi criado um primeiro exemplo com a ordem de produção do PN AAAA com 2 itens associados, como visto na Figura 42.

Figura 42 - Criação de uma ordem e 2 itens para avaliação da produção na fábrica.



Fonte: Autoria Própria.

O comportamento após a passagem correta do primeiro item da lista está na Figura 43, que é o EPC 000000000027 na estação 01. Passando por sua primeira estação, há uma mudança do estado do item que pode ser observado na Figura 44. Os passos a seguir demostra como a fábrica controla seus itens e impacta nos dados da camada da informação, com cada leitura ou estado do item alterado salvos no banco de dados:

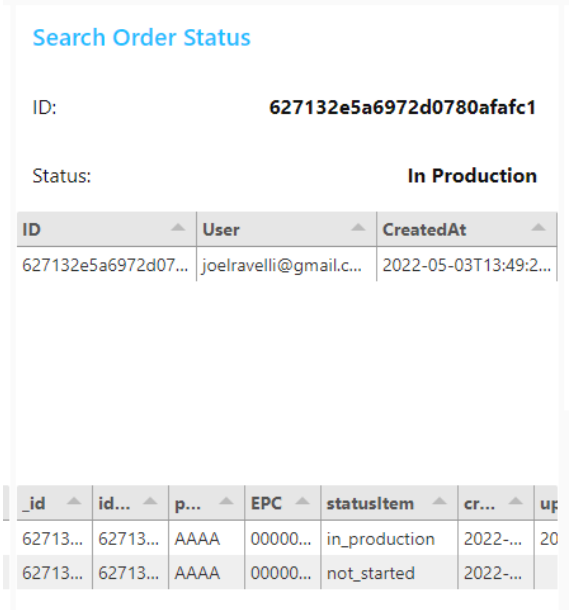
- Se nenhum item da ordem começou a ser fabricado, os itens ficam em “not_started” e a ordem fica “*Not Started*” no banco de dados. Ordens e itens são criados pela camada Funcional;
- Se um item passou por uma estação de trabalho, só este item muda para “*In Production*”, e a ordem muda para “*In Production*” também no banco de dados;

- Se um item finalizou sua produção, só este item muda para “*Finished*”, e a ordem continua em “*In Production*”, até todos os itens finalizarem a produção;
- Se todos os itens finalizarem, a ordem muda para “*Finished*” no banco de dados.

Figura 43 - EPC 000000000027 passando corretamente por todas as estações.

Fonte: Autoria Própria.

Figura 44 - O estado da ordem após finalização do primeiro item.



Fonte: Autoria Própria.

Observe na Figura 43 que o item com EPC 000000000027 faz o caminho correto, passando por todas as estações corretas para sua produção.

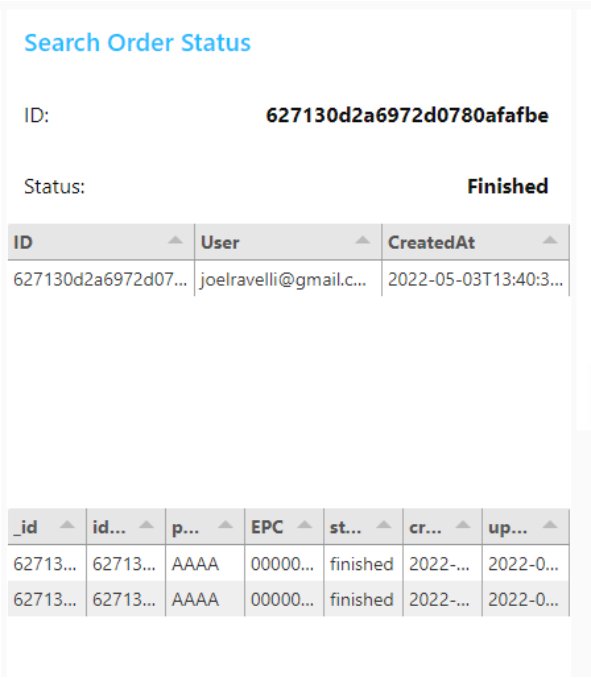
Já na Figura 45, observe que o EPC 000000000028 passou corretamente pela “Estação 01” e incorretamente pela “Estação 02”, que comete um erro de rota. Então, o semáforo acusa o erro -1 e o item não é processado nessa estação. Seguindo para a “Estação 05”, que é a próxima estação correta, finaliza corretamente sua produção.

Figura 45 - A roda do EPC 000000000028.

Fonte: Autoria Própria.

Como pode-se observar na Figura 46, onde é possível ver todos os itens com o estado “Finished” e a ordem com estado “Finished” também. Assim, temos a finalização da produção da ordem.

Figura 46 - O estado final da ordem.

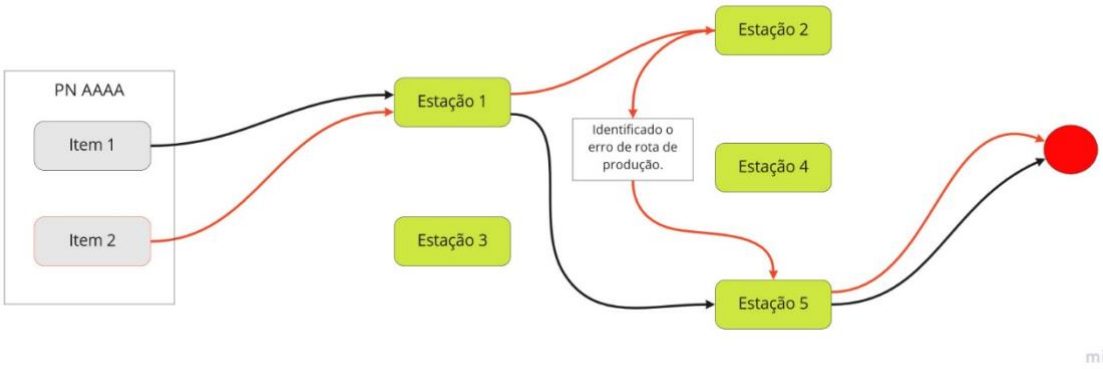


Fonte: Autoria Própria.

A Figura 47 modela o fluxo da ordem de produção com os dois itens correspondentes ao caso de teste anterior, onde é possível ver que o item 1 passou por todas as estações corretamente e já o item 2 passou corretamente pela “Estação 01” e incorretamente pela “Estação 02”, que comete um erro de rota. Então, o item não é processador nessa estação e é encaminhado para estação “Estação 05” corretamente, que é a próxima estação correta, finalizando corretamente a ordem de

produção.

Figura 47 - Ordem de produção de 2 itens e PN AAAA.



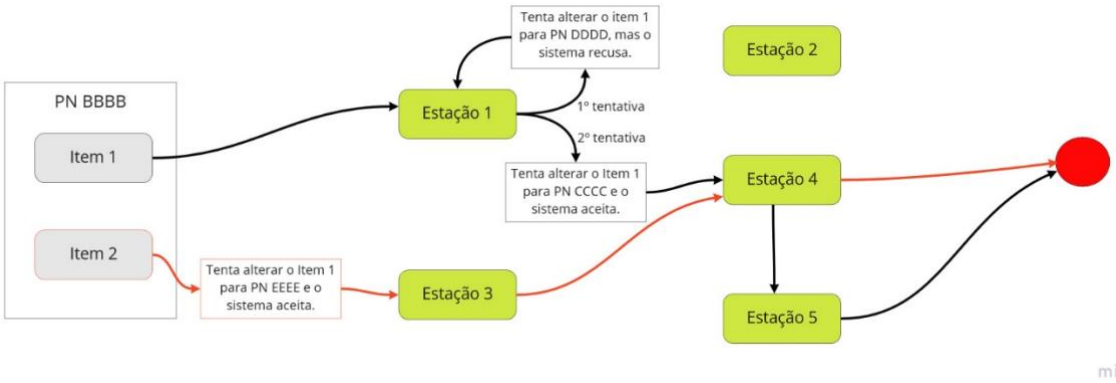
Fonte: Neto (NETO, 2022).

Um segundo exemplo é proposto para melhor entendimento do leitor, com a criação da ordem de produção utilizando o PN BBBB. O pedido também possui 2 itens e será explorada a tentativa de alteração um dos itens de um PN para outro.

O primeiro item passa pela “Estação 01” corretamente e o segundo não se iniciou. Primeiro, o cliente da ordem tenta alterar o primeiro item programaticamente para PN DDDD via interface do cliente, mas o sistema não permite e dá erro, pois a rota de produção para o PN DDDD é a passagem nas estações 2 e 5, e este item já passou pela “Estação 01”. Assim, é feita outra tentativa de alterar para o PN CCCC cuja rota de produção é 1, 4 e 5, e o sistema retorna com sucesso, pois o mesmo item já passou na “Estação 01”. A mudança não significa nenhum dano para a produção e a mudança do PN BBBB para o PN CCCC é válida, pois a “Estação 01” é comum entre os PN. Para o segundo item, o retorno é de sucesso quando se tenta alterar para PN EEEE, porque o item não iniciou sua produção e não passou por nenhuma estação, mostrando que nesta fase inicial, qualquer alteração é possível.

Este segundo exemplo mostra que um cliente arrependido pode mudar a ordem de pedido de uma mercadoria para outra, quando são observadas as regras de produção. O fluxo também é mostrado na Figura 48.

Figura 48 - Ordem de produção com 2 itens com PN BBBB.



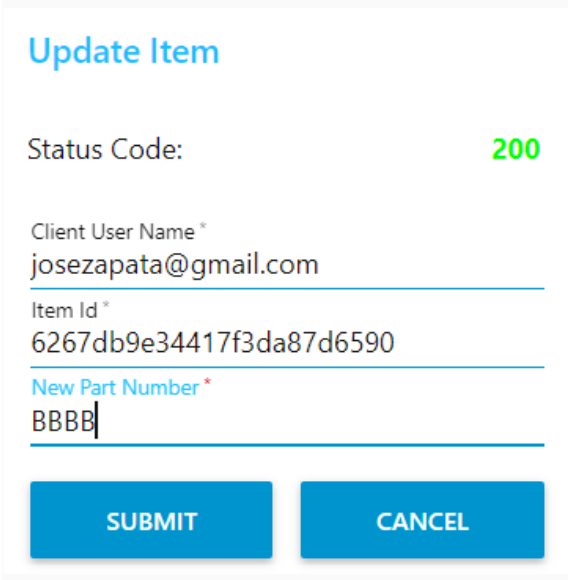
Fonte: Neto (NETO, 2022).

Para demonstrar como alterar um Item e torná-lo um bem final diferente, é possível atualizar a rota de conhecimento do item. Por exemplo, na Figura 49, um dos itens da Ordem do PN AAAA (rota 1 e 5) será transformado em PN BBBB (rota 1 e 4), o que significa que, ao invés de fazer uma camisa vermelha, uma camisa preta é feita. É possível se:

- O item não iniciou a produção;
- A rota do item atual possui semelhança das estações iniciais para habilitar a mudança, por exemplo: se o item passou pela “Estação 01”, que é semelhante de PN AAAA e BBBB, o sistema irá aceitar a mudança, porque a mudança não prejudica a construção do item final.

Caso os itens não possuam semelhanças, o sistema bloqueará a alteração para não causar danos ao item; neste caso o sistema bloqueia e retornará um erro. Isso permite uma rápida interação detalhada com a fábrica com o sistema do cliente e é a entrega central desta pesquisa, sem perder a governança da produção.

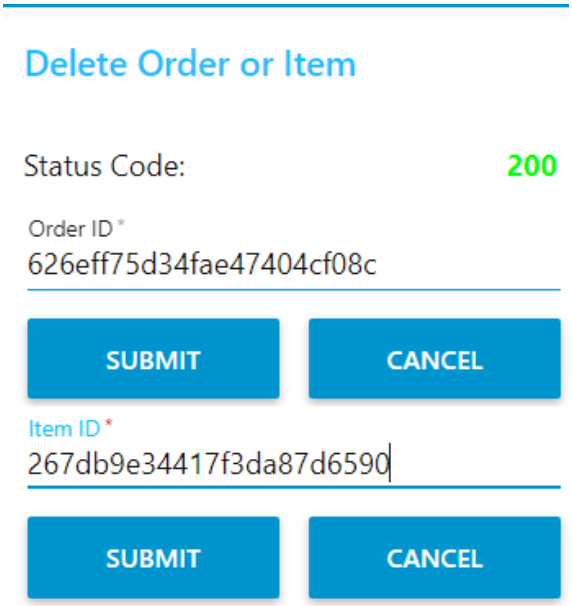
Figura 49 - Tela de atualização de itens.



Fonte: Neto (NETO, 2022).

Caso o cliente tenha perdido o interesse na produção, os itens e os pedidos podem ser excluídos, conforme mostra a Figura 50. Para isso, basta o cliente inserir o identificador do pedido e enviar para o sistema, ou inserir o identificador do item e enviar. Isso ajuda o cliente a reduzir a quantidade do pedido, se necessário. Se o cliente fez um pedido muito grande, mas deseja cancelar parcialmente ou totalmente, por qualquer motivo que possa ter, o cliente pode tentar a exclusão de itens que não se iniciaram na tela de exclusão. É possível excluir o pedido inteiro caso esteja com o estado de não iniciado (“Not Started”), mostrando que o sistema permite que o cliente tenha flexibilidade para alterar pedidos e itens conforme desejar quando a fábrica autorizar e não infringir regras de produção.

Figura 50 - Tela para deleção de Ordens e Itens.



Fonte: Neto (NETO, 2022).

5.4. Discussões e Considerações Finais

Os resultados apresentados na seção anterior possibilitam entender como as linhas de produção de uma manufatura flexíveis podem fazer uso da arquitetura de microsserviços, baseada em camadas RAMI 4.0 desenvolvidas nesta pesquisa. A arquitetura foi capaz de entregar uma maneira simples de gerenciar linhas de produção flexíveis, usando o conceito de números seriais dos produtos contidos no EPC e rota de produção, com controle baseado em serviços de gerenciamento da produção. A arquitetura demonstra que atende aos requisitos de fabricação flexível, pois os pedidos podem ser alterados durante a produção, possibilitando que os itens sejam modificados e a rota de produção possa mudar sem a necessidade de tempo extra para a reconfiguração das estações na produção. Outro ponto é a flexibilidade da arquitetura sobre a capacidade das localizações físicas das estações de trabalho e a movimentação dos itens poder mudar sem afetar a fábrica. Pode-se adicionar ou subtrair estações de produção da linha, para produzir mais ou menos produtos. As rotas das estações podem ser atualizadas no banco de dados para uma configuração mais adequada na produção de um produto específico. Em poucas palavras, tudo isso mostra os conceitos de manufatura flexível abordados nesta pesquisa.

A arquitetura entrega uma forma muito flexível e eficiente de produzir na fábrica,

entregando valor aos clientes da fábrica, que usando este recurso podem responder rapidamente às mudanças do mercado e facilmente fazer modificações em itens do seu pedido. É muito simples adaptar o sistema a diversos outros ambientes de produção, linhas de produção e diferentes cenários, devido ao uso da arquitetura de microsserviços e a modelagem da informação realizada no trabalho.

Um ponto de desvantagem desta proposta é a dependência do funcionamento correto da infraestrutura de TI, pois todos as comunicações entre a fábrica e componentes de softwares, como o servidor OPC UA e o orquestrador OPC UA, necessitam de condições boas de instalações da rede, para que não haja interrupção do tráfego de dados.

Outro ponto que é válido observar é relativo ao uso da arquitetura de microsserviços. Os programas escritos como serviços possuem a característica de estarem descentralizados na composição da arquitetura. Assim, a manutenção necessita de uma equipe de TI mais capacitada para detecção e monitoramento, reparo dos serviços distribuídos, segurança do software e comunicação entre eles.

Pode-se observar nesta pesquisa que todas as camadas do RAMI 4.0 foram utilizadas, o que raramente é identificado na literatura relacionada a esta pesquisa. O modelo de referência do RAMI 4.0 permitiu uma forma de estruturar a solução de maneira compreensível, facilitando o entendimento dos papéis e responsabilidades de cada sistema, software e ativos, em camadas do RAMI 4.0 dentro da solução de fábrica no contexto da Indústria 4.0. Uma desvantagem do RAMI 4.0 é ser um modelo abrangente e abstrato, causando divergências de interpretações.

Pode-se destacar outros pontos observados nesse trabalho e os principais objetivos alcançados foram:

- Modelagem agnóstica da informação com base no *OCG SensorThings API*, onde podemos facilmente modelar novos tipos de elementos no chão de fábrica, como obter dados de sensores e controlar atuadores diferentes por meio do hardware e software disponível na manufatura e compatível com o OPC UA. Nesse trabalho, é modelado uma manufatura têxtil inteligente, mas é possível modelar outros tipos de manufaturas com propósitos diferentes.
- Divisão e organização de código com a arquitetura em microsserviços.
- Implementar o programa *OPC UA Orchestrator* em uma arquitetura de microsserviços, que possibilita o tráfego de dados na camada da informação;

- O uso de eventos permitindo a programação assíncrona, rastreamento da informação trafegada entre serviços e resiliência com mecanismos de número de tentativas para retransmissão de dados no servidor de mensageria (*Transporter* do Moleculer);
- Todas as comunicações entre as camadas de Comunicação, Informação e Funcional estão habilitadas a utilizar a autenticação e encriptação de canal de comunicação para segurança dos dados;
- Possibilidade de realizar a instalação de serviços no ambiente em nuvem interno para maior velocidade de tráfego entre componentes de software da camada da Informação e Comunicação. Apesar de ser possível, os componentes da camada da Informação instalados em um ambiente de nuvem externa acrescentam latência na comunicação;

A proposta de arquitetura foi implementada com sucesso seguindo o modelo de referência RAMI 4.0. As demonstrações e cenários mostram que o controle da produção da fábrica pode ser entregue ao cliente parcialmente, sem perder a governança de produção e processo, permitindo eficiência e escalabilidade para manufatura flexível no contexto da Indústria 4.0.

6- CONCLUSÃO

Esta pesquisa apresentou e implementou uma arquitetura em microsserviços para aplicações de manufatura flexível na Indústria 4.0, desenvolvido com o framework Moleculer, e que envolve todas as seis camadas de TI do RAMI 4.0. Com foco na camada da Informação e modelagem de informação e banco de dados, a pesquisa mostra todos os ciclos de produção combinados sob uma perspectiva de ponta a ponta, de produção baseada em serviços.

A modelagem da informação apresentou o desenvolvimento como proposto pelo *OCG SensorThings API*, com adequações pertinentes para tornar possível a proposta do trabalho sob ótica da manufatura orientado ao cliente. Ao invés da proposta tradicional, este trabalho traz a proposta de produção orientada a produto, e implementou a flexibilidade necessária para Indústria 4.0. A modelagem da informação levou em consideração as várias fontes de dados presentes na produção fabril; e a simplificação do modelo possibilitou a modelagem da informação de forma flexível com banco de dados.

O protocolo de comunicação sugerido nas especificações do RAMI 4.0 é o OPC UA, e por este motivo, o autor desenvolveu um orquestrador para interconectar a comunicação industrial OPC UA com o framework de microsserviços Moleculer. O orquestrador OPC UA é uma entrega chave deste trabalho, e está presente na camada de Informação que é capaz de coletar e gerenciar os dados que são produzidos pelos servidores OPC UA no chão de fábrica.

O projeto também contempla a implementação da prova de conceito com a finalidade de demonstração, lado a lado com os componentes de software citados. A POC demonstra um cenário em que os clientes podem realizar pedidos, visualizar o seu estado na produção da fábrica, apagar itens ou ordens de produção e modificar itens de um PN para outro. Esta última opção é importante para demonstrar a adaptabilidade da fábrica, permitindo que o cliente controle rapidamente às mudanças de seus pedidos baseado em seu negócio, sem que a fábrica perca a governança da produção; este é um diferencial importante para o cliente. O sistema de produção precisa ser capaz de se adaptar em um mercado em constante evolução, produzindo os itens que o cliente precisa com mais rapidez e versatilidade.

A arquitetura proposta para a manufatura flexível viabilizou a interface cliente-fábrica, proporcionando maior versatilidade e interoperabilidade na camada da informação, com as demais camadas do RAMI 4.0. Este trabalho entrega um projeto completo e adaptável para trabalhar em uma manufatura flexível, entregando modelagem de dados para Indústria 4.0, uso de eventos permitindo a programação assíncrona, rastreamento da informação trafegada entre serviços, e um conjunto completo de softwares em forma de microsserviços para a fábrica; além disso, serviu como uma plataforma de produção, que pode ser controlada pelo cliente, usando os microsserviços e APIs.

7- REFERÊNCIAS

AASX Package Explorer. **Github**. Disponivel em: <https://github.com/admin-shell-io/aasx-package-explorer>. Acesso em: 28 jan. 2021.

ADOLPHS, P. et al. The Reference Architectural Model RAMI 4.0 and the Industrie 4.0 Component. **zvei.org**, 2015. Disponivel em: <https://www.zvei.org/en/subjects/industry-4-0/the-reference-architectural-model-rami-40-and-the-industrie-40-component/>. Acesso em: 26 nov. 2020.

ALLWEYER, T. **BPMN 2.0:** introduction to the standard for business process modelling. [S.l.]: [s.n.], 2016.

BASSI, L. **Industry 4.0 Hope, hype or revolution?** IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI). Modena: [s.n.]. 2017.

BECKER, K. et al. Usage of Standards in the Smart Factory Web Testbed. **SMART FACTORY WEB**, 29 jun. 2020. Disponivel em: <https://www.iiconsortium.org/pdf/Usage_of_Standards_in_Smart_Factory_Web_TB_White_Paper_2020-06-29.pdf>. Acesso em: 17 jan. 2021.

BIGHETI, J. A. **Arquitetura de Automação e Controle Orientada a Microserviços para a Indústria 4.0**. Faculdade de Engenharia da Universidade Estadual Paulista “Júlio de Mesquita Filho”. Sorocaba, p. 146. 2020.

BURKE, T. OPC Unifi ed Architecture Interoperability for Industrie 4.0 and the Internet of Things. **opcfoundation.org**, jun. 2018. Disponivel em: <https://opcfoundation.org/wp-content/uploads/2017/11/OPC-UA-Interoperability-For-Industrie4-and-IoT-EN.pdf>. Acesso em: 26 nov. 2020.

CANTELON, M. et al. **NodeJs Action Manning**. 1. ed. Nova Iorque: Manning Publications Co, 2013.

CHAUHAN, A. A Review on Various Aspects of MongoDb. **International Journal of Engineering Research & Technology (IJERT)**, v. 8, n. 5, p. 90-92, Maio 2019. ISSN ISSN: 2278-0181.

CHEN, Y.; DU, Z.; GARCÍA-ACOSTA, M. **Robot as a service in cloud computing**. IEEE Int. Symp. Serv. Oriented Syst. Eng. [S.l.]: [s.n.]. 2010. p. 151-158.

CLASSIC - OPC Foundation. **OPC Foundation**, 2019. Disponivel em: <https://opcfoundation.org/about/opc-technologies/opc-classic/>. Acesso em: 27 nov. 2020.

CONTRENAS, J. D.; GARCIA, J. I.; PASTRANA, J. D. Developing of Industry 4.0 Applications. **International Journal of Online Engineering**, v. 13, 2017.

DAMODARAN, D. B. . S. S. V. S. M. PERFORMANCE EVALUATION OF MYSQL AND MONGODB DATABASES. **International Journal on Cybernetics & Informatics (IJCI)**, v. 5, n. 2, p. 387- 394, Abril 2016.

DOCKER. **Docker.com**, 2022. Disponivel em: <https://www.docker.com/>. Acesso

em: 22 set. 2022.

DRAHOš, P. et al. **Trends in industrial communication and OPC UA**. 2018 Cybernetics & Informatics (K&I). Lazy pod Makytou: IEEE. 2018. p. 1-5.

ECKHARDT, A.; MULLER, S.; LEURS, L. **An Evaluation of the Applicability of OPC UA Publish Subscribe on Factory Automation use Cases**. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). [S.l.]: [s.n.]. 2018.

ERUVANKAI, S.; MUTHUKRISHAN, M.; MYSORE, A. K. Accelerating IIOT Adoption with OPC UA.. **Internetworking Indonesia Journa**, v. 9, n. 1, p. 3-8, 2017.

GANDOMI, A. . H. M. Beyond the hype: Big data concepts, methods, and analytics. **International Journal of Information Management**, v. 35, n. 2, p. 137 – 144, 2015.

GRÜNER, S.; PFROMMER, J.; PALM, F. RESTful Industrial Communication With OPC UA. **IEEE Transactions on Industrial Informatics**, 12, 2016. 1832-1841.

HASSANZADEH, A. . M. S. A. M. S. **Towards Effective Security Control Assignment in the Industrial Internet of Things**. IEEE World Forum on Internet of Things (WF-IoT). [S.l.]: [s.n.]. 2015. p. 6.

HISTORY - OPC Foundation. **OPC Foundation**, 2019. Disponivel em: <https://opcfoundation.org/about/opc-foundation/history/>. Acesso em: 27 nov. 2020.

HOPPE, S. There Is No Industrie 4.0 without OPC UA. **opcfoundation.org**, 2017. Disponivel em: <https://opcconnect.opcfoundation.org/2017/06/there-is-no-industrie-4-0-without-opc-ua/>. Acesso em: 27 nov. 2020.

IEC 61360-1. **IEC**, 2017. Disponivel em: <https://webstore.iec.ch/publication/28560>. Acesso em: 12 dezembro 2022.

IEC 62832-2. **IEC**, 2020. Disponivel em: <https://webstore.iec.ch/publication/60214>. Acesso em: 12 dezembro 2022.

ISO 13584-42. **ISO.org**, 2010. Disponivel em: <https://www.iso.org/standard/43423.html>. Acesso em: 12 dezembro 2022.

ISO/TS 29002-10. **ISO.org**, 2009. Disponivel em: <https://www.iso.org/standard/50774.html>. Acesso em: 12 dezembro 2022.

JAZDI, N. Cyber physical systems in the context of Industry 4.0. **2014 IEEE International Conference on Automation, Quality and Testing, Robotics**, p. 1-4, 2014.

JUNIOR, J. R. **Disertação de Mestrado em andamento**. Sorocaba, Brasil. 2021.

KAGERMANN, ; WAHLSTER, W.; HELBIG, J. Recommendations for implementing the strategic initiative INDUSTRIE 4.0. **acatech**, 2013. Disponivel em: <https://en.acatech.de/publication/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/>. Acesso em:

26 nov. 2020.

KHAKIFIROOZ, M. et al. A System Dynamic Model for Implementation of Industry 4.0. **2018 International Conference on System Science and Engineering (ICSSE)**, 2018.

KIRMSE, A. . K. F. . H. M. **Industrial big data:** From data to information to actions. Proceedings of the 4th International Conference on Internet of Things, Big Data and Security. [S.l.]: SciTePress. 2019.

KOTA, A. K.; PRABHU, D. M. Node.js: Lightweight, Event driven I/O web development. **Informatics NIC**, Janeiro 2013. ISSN 10.13140/RG.2.1.2591.9849.

LABIRINTIS, A. . J. H. V. . Challenges and opportunities with big data, v. 5, n. 12, p. 2032-2033, 2012.

LEE, E. A.; SESHIA, S. A. **Introduction to Embedded Systems:** A Cyber-Physical Systems Approach. 2ª. ed. [S.l.]: MIT Press, 2017.

LIANG, S.; HUANG, C.-Y.; KHALAFBEIGI, T. OGC SensorThings API Part 1: Sensing. **Opengeospatial.org**, 2016. Disponivel em: <https://docs.opengeospatial.org/is/15-078r6/15-078r6.html>. Acesso em: dezembro 2022.

LU, H.; ZHIFENG, Y. **Research on key technology of the address space for OPC UA Server**. 2010 2nd International Conference on Advanced Computer Control. [S.l.]: IEEE. 2010.

LYDON, B. RAMI 4.0 Reference Architectural Model for Industrie 4.0. **International Society of Automation**, 2019. Disponivel em: <https://www.isa.org/intech-home/2019/march-april/features/rami-4-0-reference-architectural-model-for-industr>. Acesso em: 2020 nov. 26.

MALVI, K. The Positive and Negative Aspects of Node.js Web App Development. **Mind Inventory**, 2018. Disponivel em: <https://www.mindinventory.com/blog/pros-and-cons-of-node-js-web-app-development>. Acesso em: 24 out. 2021.

MARR, B. What is Industry 4.0? Here's A Super Easy Explanation For Anyone. **Forbes**, 2018. Disponivel em: <https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/?sh=6dd4c27a9788>. Acesso em: 26 nov. 2020.

MARSEU, E.; KOLBERG, D.; WEYER, S. Exemplary transfer of the RAMI 4.0 Administration Shell to the SmartFactoryKL System Architecture for Industrie 4.0 Production Systems. **smartfactory.de**, 2017. Disponivel em: <https://smartfactory.de/wp-content/uploads/2017/11/SF_WhitePaper_2-1_EN-1.pdf>. Acesso em: 26 nov. 2020.

MELO, P. F. S. D. Dispositivo de Controle para a Indústria 4.0 baseado no RAMI 4.0 e OPC UA, 2020. Disponivel em: <https://repositorio.unesp.br/bitstream/handle/11449/192851/melo_pfs_me_sjbv.pdf>. Acesso em: 25 nov. 2020.

MELO, P. F. S. et al. Open Source Control Device for Industry 4.0 Based on RAMI 4.0. **Electronics 2021**, 2021. Disponivel em: <https://repositorio.unesp.br/handle/11449/192851>.

MOLECULER TRANSPORTER. Transporter. **Moleculer Services**, 2021. Disponivel em: <https://moleculer.services/docs/0.12/transporters.html>. Acesso em: 16 Março 2021.

MOLECULERJS. What is Moleculer? **Moleculer**, 2021. Disponivel em: <https://moleculer.services/docs/0.14/>. Acesso em: 15 Março 2021.

MONGODB Features. **mongodb**, 2021. Disponivel em: <https://docs.mongodb.com/manual/introduction/>. Acesso em: 15 Março 2021.

MONGODB. MongoDB Atlas. Fully managed MongoDB in the cloud. Disponivel em: <https://www.mongodb.com/>. Acesso em: 22 set. 2022.

NETO, J. Z. Development of a Business Interface based on RAMI 4.0 for Flexible Manufacturing in Industry 4.0, 2022.

NIKOLAIDIS, P. et al. **Applying Mitigation Mechanisms for Cloud-based Controllers in Industrial IoT Applications**. Malardalen Real-Time Research Centre (MRTC), Malardalen University. Suécia. 2015.

NOVAK, P. . V. J. . K. P. . K. L. . K. . W. D. . B. S. **Engineering Roles and Information Modeling**. IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). Zaragoza, Espanha: IEEE. 2019.

PEREIRA, C. R. **Aplicações web real-time com Node.js**. 1. ed. São Paulo: Casa do Código, v. 1, 2014.

PONTAROLLI, R. P. . B. J. A. . F. M. M. . D. F. O. . R. S. L. . G. E. P. **Microservice Orchestration for Process Control in Industry 4.0**. Metrology for Industry 4.0 & IoT 2020 IEEE International Workshop. Roma, Italia: IEEE. 2020. p. 245-249.

RAHIMI, H.; ZIBAEENEJAD, A.; SAFAVI, A. A Novel IoT Architecture based on 5G-IoT and Next Generation Technologies, Vancouver, 2018.

REFERENCE Architecture Model Industrie 4.0 (RAMI4.0). **Beuth**, 2016. ISSN DIN SPEC 91345. Disponivel em: <https://www.beuth.de/en/technical-rule/din-spec-91345/250940128>. Acesso em: 28 jan. 2021.

ROBIN, A. JSON Encoding Rules SWE Common / SensorML. **OGC**, 18 jan. 2018. Disponivel em: <http://docs.opengeospatial.org/bp/17-011r2/17-011r2.html>. Acesso em: 17 jan. 2021.

ROBO3T - ROBOMONG. Robo 3T is now Studio 3T Free. Disponivel em: <https://robomongo.org/>. Acesso em: 22 set. 2022.

RÜTTIMANN, B. G. Lean and Industry 4.0 - Twins, Partners, or Contenders? A Due Clarification Regarding the Supposed Clash of Two Production Systems. **Journal of Service Science and Management**, v. 09, n. 06, p. 485-500, 2016.

SANISLAV, T.; LIVIU, M. Cyber-physical systems - Concept, challenges and research areas. **Control Engineering and Applied Informatics 14**, 2012. 28–33.

SCHROTH, C.; JANNER, T. Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services, v. 9, n. 3, p. 36-41, 2007.

SOOS, G. . F. D. . E. V. P. **Investigating the network traffic of Industry 4.0 applications - methodology and initial results**. International Conference on Network and Service Management (CNSM). Izmir, Turquia: IEEE. 2020.

STATISTA. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025, 2016. Disponivel em: <https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/>. Acesso em: 04 nov. 2020.

SURI, K. et al. **Modeling business motivation and underlying processes for RAMI 4.0-aligned cyber-physical production systems**. 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA). [S.l.]: [s.n.]. 2017. p. 1-6.

SURI, K. et al. Modeling business motivation and underlying processes for RAMI 4.0-aligned cyber-physical production systems. **22nd IEEE International Conference on Emerging Technologies and Factory Automation**, 2017.

TELLO, D.; EDGAR, G. Industry 4.0: Application of advanced services in logistics, 2018. Disponivel em: <https://upcommons.upc.edu/bitstream/handle/2117/121967/memoria_revisada.pdf>.

UNIFIED Architecture - OPC Foundation. **opcfoundation.org**, 2019. Disponivel em: <https://opcfoundation.org/about/opc-technologies/opc-ua/>. Acesso em: 26 nov. 2020.

VELOSA, A. . K. D. L. B. . W. R. Hype Cycle for the Internet of Things, 2020, 2020. Disponivel em: <https://www.gartner.com/en/documents/3987602/hype-cycle-for-the-internet-of-things-2020>. Acesso em: 07 Outubro 2020.

VYATKIN, V. Software engineering in industrial automation: State-of-theart review. **IEEE Transactions on Industrial Informatics**, v. 9, n. 3, p. 1234–1249, 2013.

WANG, Y.; TOWARA, T.; ANDERL, R. Topological Approach for Mapping Technologies in Reference Architectural Model Industrie 4.0 (RAMI 4.0). **Proceedings of the World Congress on Engineering and Computer Science**, San Francisco, v. II, 2017.

WITKOWSKI, K. **Internet of Things, Big Data, Industry 4.0 – Innovative Solutions in Logistics and Supply Chains Management**. 7th International Conference on Engineering, Project, and Production Management. [S.l.]: [s.n.]. dez. 2017.

WOLFGANG, M.; STEFAN-HELMUT, L.; MATTHIAS, D. **OPC Unified Architecture**. [S.l.]: Springer, 2009. ISBN ISBN 978-3-540-68899-0.

WU D., G. M. . R. D.; D., S. Cloud manufacturing: Strategic vision and state-of-the-art. **J. Manuf. Syst.**, v. 32', n. 4, p. 564-579, 2013.

YE, X.; HO HONG, S. **An AutomationML/OPC UA-based Industry 4.0 Solution for a Manufacturing System**. 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). [S.l.]: [s.n.]. 2018.

ZEZULKA, F. et al. Industry 4.0 – An Introduction in the phenomenon. **14th IFAC Conference on Programmable Devices and Embedded Systems PDES**, v. 49, p. 8-12, 2016.

ZEZULKA, F. et al. **Communication Systems for Industry 4.0 and the IIoT**. 15th IFAC Conference on Programmable Devices and Embedded Systems. Ostrava: [s.n.]. 2018. p. 150-155.