UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
CAMPUS DE ILHA SOLTEIRA

MATEUS DE OLIVEIRA SILVA

HARMONIZAÇÃO DO MODELO DE INFORMAÇÃO COMUM E MULTISPEAK PARA O GERENCIAMENTO DE ENERGIA DAS CASAS INTELIGENTES

Ilha Solteira - SP
2024

MATEUS DE OLIVEIRA SILVA

HARMONIZAÇÃO DO MODELO DE INFORMAÇÃO COMUM E MULTISPEAK PARA O GERENCIAMENTO DE ENERGIA DAS CASAS INTELIGENTES

Trabalho de Graduação apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Jonatas Boas Leite

Ilha Solteira - SP

2024

FICHA CATALOGRÁFICA

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

S586h

Silva, Mateus de Oliveira.
Harmonização do modelo de informação comum e Multispeak para o gerenciamento de energia das casas inteligentes / Mateus de Oliveira Silva. -- Ilha Solteira: [s.n.], 2024
53 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista (UNESP), Faculdade de Engenharia, Ilha Solteira, 2024

Orientador: Jonatas Boas Leite

Inclui bibliografia

1. Clientes. 2. Central de operações. 3. Harmonização. 4. Modelo de Informação comum. 5. Multispeak.

Elaborado por Raiane da Silva Santos - CRB - 8/9999

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte e seis dias do mês de junho do ano de dois mil e vinte e quatro, o discente *Mateus de Oliveira Silva*, matriculado sob o nº 162053576, tendo como banca examinadora o seu orientador, *o Prof. Dr. Jonatas Boas Leite, o Prof. Dr. Carlos Antonio Alves e o Dr. Lucas do Carmo Yamaguti*, apresentou o Trabalho de Graduação intitulado **"Harmonização do Modelo de Informação Comum e *MultiSpeak* para o Gerenciamento de Energia das Casas Inteligentes"**, obtendo a nota 8,0 (OITO) e conceito APROVADO.

Prof. Dr. Jonatas Boas Leite
- Orientador-

Mateus de Oliveira Silva
- Discente-

Prof. Dr. Carlos Antonio Alves
- Membro da Banca -

Dr. Lucas do Carmo Yamaguti
- Membro da Banca -

Faculdade de Engenharia – Departamento de Engenharia Elétrica
Avenida Professor José Carlos Rossi, 1370 - CEP 15.385-000, Ilha Solteira - SP - Brasil
Tel (18) 37431150 - Fax (18) 3742-2735 - www.dee.unesp.br

AGRADECIMENTOS

Dedico este trabalho, principalmente, aos meus pais, foi pelo seu apoio emocional e financeiro que tudo isso foi possível; ao meu orientador pela dedicação e qualidade de ensino, e aos meus amigos.

“As universidades serão o que são suas bibliotecas” (Gelfand, 1968, p. 19, tradução nossa).

RESUMO

Utilizando o conceito de *Smart Grids* com foco na interoperabilidade de dados entre o cliente e a central de operações, tem-se como objetivo principal do trabalho a elaboração de um modelo de compatibilidade de dados entre o *Home Energy Management System* (HEMS) e o sistema de gerenciamento da distribuição. O HEMS é desenvolvido segundo as características de um sistema supervisório, permitindo ao cliente de energia elétrica monitorar e controlar as cargas elétricas residenciais utilizando um computador pessoal com uma interface gráfica de fácil compreensão e interativa. Assim, as empresas distribuidoras de energia elétrica podem utilizar esta tecnologia para aumentar a qualidade de energia entregue aos consumidores e a confiabilidade de suas redes elétricas. A viabilidade dessa troca de informações exige um modelo de informação compatível com o domínio do cliente e com o domínio da concessionária distribuidora de energia elétrica. Da utilização do *Multispeak* no domínio de pequenas cooperativas e do Modelo de Informação Comum (CIM) no domínio das grandes concessionárias, esse trabalho busca desenvolver um modelo de dados harmonizado entre CIM e *Multispeak*. Os resultados ilustram as bibliotecas utilizadas, os tipos básicos e adaptações realizados na harmonização. Os códigos são apresentados em linguagem de programação com o auxílio dos softwares *Visual Code Studio, replit, Enterprise Architect,* em C#, além de diagramas de classes padronizadas. Não houveram custos adicionais na realização desse trabalho, todas as ferramentas utilizadas foram de livre acesso.

Palavras-chave: clientes, central de operações, harmonização; modelo de informação comum; *multispeak*.

ABSTRACT

Utilizing the concept of Smart Grids with focus on data interoperability between customer and the operations center, the main objective of the work is to develop a data compatibility model between the Home Energy Management System (HEMS) and the Distribution Management System. The HEMS is developed according to the characteristics of a supervisory system, allowing the electricity customer to mon- itor and control residential electrical loads through a personal computer, using an easy- to-understand and interactive graphical interface. Thus, electricity distribution companies can use this technology to increase the quality of energy delivered to consumers and the reliability of their electrical networks. The viability of this information exchange requires an information model compatible with the customer's domain and the domain of the electricity distribution concessionaire. Using Multispeak in the domain of small cooperatives and the Common Information Model (CIM) in the domain of large concessionaires, this work seeks to develop a harmonized data model between CIM and Multispeak. The results demonstrate the libraries used, the basic types and adaptations made in the harmonization. The codes are presented in programming language with the aid of Visual Code Studio, replit, Enterprise Architect, in C#, in addition to standardized class diagrams. There were no additional costs in the develepment of this work, all tools were free access.

Keywords: customers, operations center, harmonization; common information model; multispeak.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CIM	*Common Information Model*
DMS	*Document Management Systems*
EMS	*Enterprise Messaging System*
ESB	*Enterprise Service Bus*
HEMS	*Home Energy Management System*
IdC	Internet das Coisas
IEC	*International Electrotechnical Commission*
JSON	*JavaScript Object Notation*
MQTT	*Message Queuing Telemetry Transport*
NIST	*National Institute of Standards and Technology*
NRECA	*National Rural Electric Cooperative Association*
OT	*Operational Technology*
RDF	*Resource Description Framework*
SCADA	*Supervisory Control and Data Acquisition*
SCL	*Structure Control Language*
SOAP	*Simple Object Access Protocol*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A *Web* das Coisas permite às aplicações de *software* utilizar dados de sensores, atuadores e demais fontes de dados, aumentando a intercomunicação entre dispositivos. Devido ao surgimento de arquiteturas relacionadas a Internet das Coisas (IdC), surge a necessidade de interligar as operações em um conjunto comum padronizado de alto nível capaz de criar, manipular e permitir a configuração de *hiperlinks* individuais das máquinas, montados em respectivos protocolos de *web*. Para tal, modelos de informação podem ser baseados em reutilização de informação, atributos e relações. A IdC dá acesso aos dispositivos pelo uso de redes e protocolos de Internet, flexível a ponto de adaptar a atual e futuras redes de sensores (EPRI, 2012).

Dentro do estabelecido surge o Modelo de Informação Comum, do Inglês, *Common Information Model* (CIM), um padrão para distribuição e transmissão de energia elétrica que permite às aplicações de *softwares* a troca de informação sobre a rede elétrica, oficialmente adotado pela Comissão Eletrotécnica Internacional, do Inglês, *International Electrotechnical Commission* (IEC) (EPRI, 2012).

O CIM é definido por meio da Linguagem de Modelagem Unificada, do Inglês, *Unified Modeling Language* (UML), que descreve uma linguagem básica e comum de ontologia. Apresenta a capacidade de moldar a rede em si usando modelo de classes, descrevendo os componentes básicos utilizados no transporte de eletricidade.

O CIM também oferece a capacidade de criar mensagens para o comércio de energia com estrutura para comunicações relacionadas ao mercado de energia, padronizado pela IEC 62325. O perfil europeu de mercado é fundamentado no CIM, objetivando harmonizar as trocas de dados do mercado de energia no continente.

Os fundamentos do CIM são definidas na parte IEC 61970-301, com foco na transmissão de eletricidade, além do sistema supervisório de controle e aquisição de dados, do Inglês, *Supervisory Control and Data Acquisition* (SCADA), planejamento e otimização. As partes IEC 61970-501 e 61970-452 definem um formato *Extensible Markup Language* (XML) para trocas de dados de rede usando *Resource Description Framework* (RDF) (*Power Europe,* 2023). A série de padrões IEC 61968 possibilita que o CIM obtenha os meios necessários para a distribuição de eletricidade, incluindo sistema de gerenciamento de distribuição, sistema de gerenciamento de interrupção,

planejamento, medição, sistema de informação geográfica, gerenciamento de trabalho e ativos, sistema de informação do consumidor e planejamento de recursos empresariais (Simmins, 2011).

O conceito de harmonização surge para melhorar a interoperabilidade e troca de dados pela transformação de modelos, como *Structure Control Language* (SCL) em CIM. Isso permite o desenvolvimento de aplicação e sistemas gerais sem que sejam moldadas para apenas uma implementação em particular. Para o caso de harmonização com o *MultiSpeak*, são identificados os níveis de correlação (onde os modelos se alinham) existentes entre CIM, baseado nos padrões IEC 61968, 61970, e 62325, e o padrão *MultiSpeak* para interoperabilidade de sistemas elétricos. São mapeados os atributos dos perfis CIM equivalentes que são criados para o padrão *MultiSpeak*, logo não se busca resolver as diferenças, mas sim em encontrar as similaridades de como eles podem ser usados juntos (EPRI, 2012).

O *MultiSpeak* foi fundado *pelo National Rural Electric Cooperative Association* (NRECA) e escolhido como padrão essencial nas operações de rede do Modelo Conceitual da *Smart Grid* desenvolvida pelo *National Institute of Standards and Technology* (NIST) (NIST, 2010). É o padrão de integração mais amplamente adotado na América do Norte, nas centrais de operação de distribuição das concessionárias. Apresenta flexibilidade, na qual vendedores que se tornaram membros contribuem com suas experiências no aperfeiçoamento do padrão (*MultiSpeak*, 2024).

Três componentes chave definem o *MultiSpeak*: 1) detalhes dos dados compartilhados são documentados em esquemáticos XML; 2) estruturas de mensagem que suportam troca de dados e serviços de web com estrutura para troca em tempo real; 3) além de detalhamento dos processos de negócios com passo a passo para realizar os dados (*MultiSpeak*, 2024).

Ambos os padrões *MultiSpeak* e CIM servem no domínio de integração de aplicativos das concessionárias, mas com diferentes propósitos e estrutura básica. Para o caso do CIM, são atendidas as concessionárias de grandes investimentos, com a ideia de equipes robustas capazes de manusear grandes fluxos de informação e moldando o padrão para suas respectivas necessidades. Por sua vez, *MultiSpeak* atende serviços menores como municípios e cooperativas com equipes de tecnologia da informação mais limitados; *MultiSpeak* começou como um padrão de interface para depois se tornar uma linguagem de modelo unificado. O problema surge, então, ao se deparar com a necessidade de integrar dois sistemas distintos em conformidade com

padrões diferentes. Como resolvem desafios diferentes, cria-se no mercado o conflito de qual padrão suportar, elevação dos preços de integração, como suportar ambos, e como integrar os sistemas baseados em padrões diferentes (EPRI, 2012).

Portanto, CIM cobre internacionalmente a transmissão, geração e distribuição, em especial na área das concessionárias, enquanto *MultiSpeak* é focado na etapa de distribuição e de atender as necessidades das cooperativas de energia elétrica, principalmente nos Estados Unidos. A IEC 61968 não depende de protocolos de transporte, sendo o oposto do *MultiSpeak* (nesse caso são usados protocolo de mensagens, ou *Simple Object Access Protocol* (SOAP), em *sockets* HTTP, TCP/IP diretamente entre aplicações e transferências baseadas em arquivos para o transporte de dados). O esquemático XML é usado por ambos ospadrões para definição de mensagens.

1.1. OBJETIVOS

Este trabalho de graduação tem como objetivo principal o desenvolvimento de uma interface de compatibilidade entre os modelos CIM e *MultiSpeak* para padronizar as trocas de informações entre o sistema de gerenciamento da distribuição, do Inglês, *Distribution Management System* (DMS) e o sistema de gerenciamento de energia residencial, HEMS.

O método apresentado neste trabalho de graduação engloba o uso da *Sparx-Systems Enterprise Architect* para exportar os perfis necessários ao funcionamento do modelo. O processo inicial utiliza a obtenção do perfil de *MeterReading* para encontrar as classes ramificadas na árvore. Para a escrita do código é utilizado o site *replit,* na linguagem C#, em conjunto com o *software Visual Studio Code*.

1.2. ESTRUTURA DO TEXTO

O texto está organizado da seguinte forma. Neste capítulo 1 há uma introdução ao tema desenvolvido no trabalho de graduação com a definição dos objetivos. No Capítulo 2 é realizado um estudo mais aprofundado sobre os padrões CIM e Multispeak. No Capítulo 3 são apresentados alguns conceitos sobre a UML, a adaptação CIM e Multispeak e a arquitetura IdC multicamadas. No Capítulo 4 são apresentados os resultados com os códigos dos modelos desenvolvidos em linguagem de programação. Por fim, têm-se as conclusões do trabalho.

2 CIM E *MULTISPEAK* PARA CONSUMIDORES DE ENERGIA

As empresas concessionárias de energia estão a utilizar e idealizar os Sistemas de Gerenciamento dos Recursos Energéticos Distribuídos, do Inglês, DERMS, Recursos Energéticos Distribuídos, do Inglês, DER, e outras tecnologias de suporte à transição energé- tica, para mitigação das alterações climáticas com metas de sustentabilidade. Isto exige um sistema interoperável com aplicações de intercâmbio de dados entre múlti- plas partes do setor elétrico, tais como, concessionárias de distribuição, geradores, ope- radores de rede, reguladores e entidades governamentais. A emergência de uma rede verdadeiramente interligada e digital é clara em todo o mundo. Exemplos na indústriahoje incluem:

- As soluções de DERMS das concessionárias de energia elétrica fornecidas por agentes do mercado aos consumidores de DER renovável, interconectado e integrado com o DERMS. Essas soluções estão sendo usadas atualmente em sistemas do mercado de energia, incluindo o Operador Independente do Sistema da Califórnia (CAISO) (Mannheim & Malekos, 2020) e o Operador de Sistema Independente do Centro-Oeste (MISO) (Swenson, 2021).
- O consórcio de empresas concessionárias de energia elétrica na Irlanda para criar a interligação da Irlanda com a França, bem como a integração na rede nacional do Reino Unido. Estes novos mercados são alimentados pela interligação dos sistemas DER e DERMS da Irlanda para estações de carregamento fotovoltaico (PV), eólico e de veículos elétricos (EV) na ilha.

Os modelos padrão existentes podem ser adaptados para definir modelos de informações padrão da indústria de software que usam tecnologias como JSON, o que ajuda a criar uma versão *JavaScript Object Notation* (JSON) de micros-serviços RESTful na indústria de energia elétrica. Esses padrões estendidos de integração do setor de serviços públicos podem ser usados em sistemas de tecnologia operacional, do Inglês, OT, como DERMS, e diretamente com DER. Outros sistemas de OT onde a nova tecnologia de integração beneficiaria são: sistemas de gerencia- mento de energia; sistemas de gerenciamento de mercado; e plataformas emergentes

de gerenciamento de contingências. Os ecossistemas de sistemas da concessionária também podem usar padrões semelhantes para fins de aplicação e integração.

O CIM apenas define o modelo, no entanto, para que o CIM seja empregado de forma mais ampla, deve-se definir o modo de implementar o CIM como um serviço padrão. Como implementador do CIM seria muito melhor ter serviços definidos em um padrão, o que facilitaria a implementação do padrão. Para o desenvolvedor de *software*, o *MultiSpeak* é frequentemente preferível ao CIM devido à facilidade de implementação dos serviços *MultiSpeak*. Se a CIM definisse serviços padrão a serem implementados, isso se tornaria mais amplamente aceito no setor elétrico.

A flexibilidade dos padrões CIM *e MultiSpeak* permite que esses padrões sejam facilmente estendidos devido às definições de seus modelos. No entanto, novas ferramentas e tecnologias são necessárias para ajudar os fornecedores a ampliar esses padrões e criar padrões mais leves, para criar modelos de informação, como JSON, que podem ser implementados como serviços RESTful.

Recomenda-se ter estas ferramentas de padrões da indústria como *plug-ins* dispo- níveis no mercado para que os fornecedores de tecnologia na indústria de energia usassem e implementassem modelos de informação padrão interoperáveis e arquite- turas de serviços para *software* que fosse compatível e interoperável. Isso também deve ter capacidade de teste e certificação para empresas que optem por implementarmodelos padrão do setor e serviços REST (Mullenmaster et al., 2022).

Ao adotar uma arquitetura orientada a serviços, tanto o *MultiSpeak* quanto o CIM atendem às demandas de interoperabilidade das aplicações empresariais necessárias para atingir as metas de interoperabilidade do setor de energia. Ambos os padrões permitem qualidade e interoperabilidade de *software* mais rápidas e historicamente melhores. Por serem modelos padrão, eles possuem ferramentas para que os fornecedores de desenvolvimento de *software* implementem o padrão. Os padrões CIM e *MultiSpeak* e as partes envolvidas associadas procuram alcançar uma interoperabilidade perfeita dos sistemas e soluções da indústria de energia.

O CIM cobre transmissão, geração e distribuição, enquanto o MultiSpeak é focado na distribuição. Além disso, a *MultiSpeak* originalmente procurou atender às necessidades das cooperativas elétricas nos EUA, enquanto o CIM esteve mais focado em todas as concessionárias do mercado internacional. Com o passar do tempo, am-

bos os padrões evoluíram para atender uma parcela mais ampla do mercado da indústria de energia para o uso correto dos dados sem a necessidade de interfaces de manutenção personalizadas.

2.1 CIM

O Comitê Técnico 57 (TC57) da IEC tem desenvolvido um padrão para integração de software às concessionárias. O padrão do TC57 é baseado em um modelo de objeto denominado CIM, que está documentado na norma IEC 61970-301. O CIM foi originalmente desenvolvido para suportar centros de transmissão e controle, mas tem sido ampliado para abordar todos os aspectos de uma concessionária elétrica verticalmente integrada. Como resultado, o CIM cobre um campo mais amplo do que o *MultiSpeak*, uma vez que atualmente o *MultiSpeak* trata apenas da distribuição.

O Grupo de Trabalho 14 (GT14) do TC57 concentra-se em aspectos de distribuição para o CIM. As principais extensões do CIM para resolver problemas de distribuição estão documentadas na série de padrões IEC 61968. Em particular a parte IEC 61968-1 descreve a arquitetura básica e a estrutura de mensagens para interfaces de distribuição, e a parte IEC 61968-11 descreve o modelo de informação para interfaces de distribuição. Como ambos abordam questões de distribuição, há uma sobreposição conceitual substancial no *MultiSpeak* e nos padrões 61968.

Os padrões CIM concentram-se na definição e no conteúdo da mensagem, deixando grande parte do transporte e do *middleware* como questões de implementação. Tal abordagem pode ser inadequada para as concessionárias de menor escala e para os fornecedores de *software* que servem o mercado cooperativo, muitos dos quais também têm recursos limitados. Como resultado, o *MultiSpeak* padroniza os serviços da Web como meio de transporte de dados e não pressupõe a existência de uma infraestrutura de *middleware* de mensagens.

2.2. INTEROPERABILIDADE CIM-MULTISPEAK

Um dos objetivos do *GridWise* que está claramente elucidado na estrutura de interoperabilidade é a capacidade de “fazer pontes entre comunidades com entendimentos evoluídos de forma independente”. Há uma necessidade clara de construir

uma ponte semântica entre o *MultiSpeak* e o 61968 CIM para que eventualmente seja possível promover a interoperação entre as aplicações *MultiSpeak* e CIM.

A abordagem ideal seria usar ferramentas emergentes de representação semântica para fornecer uma tradução dinâmica entre *MultiSpeak* e CIM. A comunidade CIM começou recentemente a publicar CIM no formato *Web Ontology Language* (OWL). A *MultiSpeak* atualmente está considerando dar esse passo. Acredita-se que eventualmente podem ser disponibilizadas ferramentas para facilitar a tradução eletrônica entre modelos semânticos expressos em formato OWL; no entanto, essas ferramentas ainda são muito rececntes.

Uma abordagem de curto prazo consiste no desenvolvimento de um adaptador de tradução. Desde que um mapeamento pode ser gerado entre os dois modelos de dados, as mensagens criadas usando qualquer um dos padrões poderia ser convertido eletronicamente na mensagem correspondente gerada pelo outro padrão. Ou seja, seria gerada uma tabela de tradução que indica quais dados em uma mensagem *MultiSpeak* correspondem a quais itens em uma mensagem CIM e vice-versa.

Esta solução é viável e consistente com as abordagens já adotadas pelos respectivos grupos, em que o método usado pelo *MultiSpeak* para permitir que dois programas de *software* compatíveis troquem dados, através de um “tradutor” *MultiSpeak* oferecido pelo fornecedor sem afetar os bancos de dados nativos de cada *software*. Muitas das implementações CIM até o momento usam uma camada adaptadora para integrar um aplicativo legado com aplicativos CIM.

Conceitualmente, uma tradução pode funcionar entre uma aplicação compatível com *MultiSpeak* e uma aplicação compatível com CIM. Em ambos os casos, o adaptador traduz a saída do aplicativo para corresponder ao formato esperado por outros aplicativos na rede. Há casos em que relativamente poucos aplicativos habilitados para *MultiSpeak* devem ser integrados em uma rede empresarial predominantemente baseada em CIM e outros casos em que relativamente poucos aplicativos baseados em CIM se integrariam a uma empresa baseada em rede *MultiSpeak*.

A criação dos adaptadores apropriados exige vários passos: primeiro, um mapeamento conceitual entre os dois modelos de dados e, segundo o desenvolvimento de uma tradução eletrônica utilizando esse mapeamento conceitual. Esses esforços começaram tanto no *MultiSpeak* quanto no WG14 para dar os primeiros passos para desenvolver o mapeamento conceitual. Há uma previsão de que a criação de uma

tradução de uma folha de estilo XML seja simples assim que o mapeamento conceitual for concluído.

2.2 CIM-MULTISPEAK PARA CLIENTES DE ENERGIA

O mercado de Infraestruturas de Medição Avançada, do Inglês, *Advanced Metering Infrastructure* (AMI) é relativamente imaturo e as tecnologias evoluem rapidamente – e provavelmente continuarão mudando a um ritmo acelerado. Como forma de mitigar os riscos técnicos e reduzir os custos do ciclo de vida, os fornecedores de *software* estão ativamente envolvidos e fazendo contribuições significativas para os esforços de padronização da indústria que impactam o programa de AMI. Os fornece- dores não estão apenas realizando avaliações completas de tecnologias-chave, mas estão considerando como cada tecnologia e os dados se encaixam na infraestrutura de integração geral – desde a interface com a rede de área residencial, do Inglês *Home Area Network* (HAN) até o sistema de unificação de dados do medidor, do In- glês, *Meter Data Unification System* (MDUS) e depois através de aplicativos que su- portam processos de negócios. Para conseguir economias de escala, este intercâmbio de dados deve basear-se em padrões industriais que sejam adequados e apoiados no mercado.

Através da Equipe de Definições de Serviço da Força-Tarefa AMI-*Enterprise*, os fornecedores de software estão fazendo um esforço para examinar os padrões CIM e *MultiSpeak*, determinar lacunas e fazer recomendações tanto para o MultiSpeak quanto para o IEC. Como este trabalho está sendo feito em colaboração com concessionárias e fornecedores e sendo testado como parte de seu programa AMI para implementação, os artefatos provenientes desta equipe são de alta qualidade e inestimáveis no desenvolvimento da série de padrões IEC 61968-14 acima mencionada. Consequentemente, estas normas IEC já estão ganhando um impulso substancial enquanto ainda estão em estado de rascunho (McNaughton et al., 2008).

Para identificar os requisitos de integração de aplicativos, os desenvolvedores começam com casos de uso que documentam os processos de negócios relacionados à AMI. Os requisitos de integração são identificados onde os objetos de informação são baseados de sistema para sistema, por exemplo, se a informação for passada de um Sistema de Informação do Cliente, do Inglês *Client Information System* (CIS) para

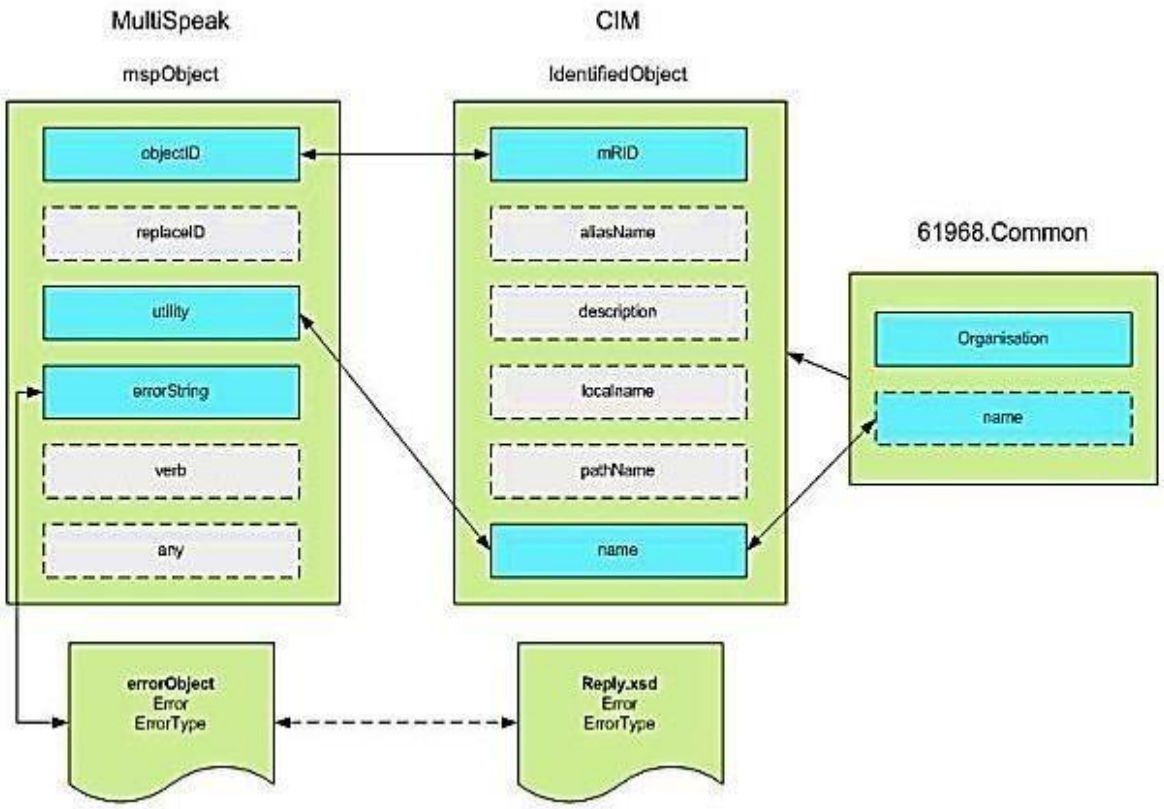
um sistema de gerenciamento de dados de medidor. Neste processo, há a necessidade de uma abordagem de nomenclatura de serviço que usa um verbo baseado em CIM, depois um objeto de informação, seguido pelo nome do padrão. Então, é construido o conteúdo de cada objeto de informação revisando CIM, *MultiSpeak* ou outras fontes de fornecedores que concordaram em compartilhar informações.

3 DESENVOLVIMENTO

3.1 PERFIS EQUIVALENTES *MULTISPEAK*-CIM

Os modelos CIM e *MultiSpeak* usam uma classe que outras herdam que também contém um identificador universal usado nos respectivos modelos (EPRI, 2012). A Figura 1 mostra uma classe de alto nível e como ela é modificada entre os dois padrões.

Figura 1 – Exemplo de mapeamento de uma classe de alto nível.



Fonte: (EPRI, 2012).

Na Figura 1 é representada a classe de alto nível *IdentifiedObject* do padrão CIM e como ela é transformada em *mspObject* ao se fazer o mapeamento em *MultiSpeak*. Nesse caso, as linhas pontilhadas representam os atributos opcionais, enquanto as contínuas os obrigatórios. No procedimento há um identificador universal, *Master Resource* ID (mRID), CIM, e *objectID* para *MultiSpeak*. Nesses casos o mapeamento

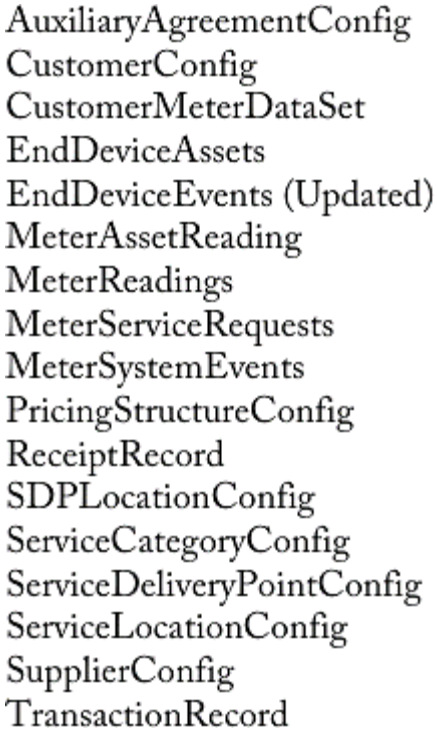
é mais direto, ao contrário de, por exemplo, “*description*”, que já representa um mapeamento opcional para o modelo CIM; no caso do *MultiSpeak*, um exemplo semelhante seria “*any*”. Esses atributos por serem opcionais podem ser ignorados no processo de harmonização. Ainda com base na Figura 1, é observado que *MultiSpeak* necessita do uso do nome da concessionária, visto que ela é usada para identificar os dados para qual concessionária são necessários e pela troca comum de informações entre as constituintes as quais o modelo é utilizado. Para o modelo CIM, esse não é um procedimento comum, porém pode acomodar a mesma função pelo uso da classe *Organization* com o atributo herdado “*name*”. Para situações em que surgem erros, como *errorString*, um atributo remanescente de versões antigas dos modelos que ainda permanecem, é possível utilizar o recurso de chamadas de serviço web, onde é mapeado para a chamada de serviço web subsequente do modelo objetivo.

Os modelos podem apresentar correlações entre si (Tabela 2), onde há alinhamento das informações, transformações, nas quais atributos podem ou não precisar de alterações (Tabela 3) ou ainda lacunas, relações entre classes equivalentes ou atributos podem não existir (Tabela 4).

O ponto de partida para o desenvolvimento do trabalho é obter a classe inicial de escolha, para dela encontrar as demais classes e todas entre elas relacionadas. No *Enterprise Architect* é possível encontrar em sua biblioteca as classes necessárias, utilizando-as para montar o perfil desejado. Como o trabalho segue a linguagem C#, é necessário primeiro a conversão, visto que o software trabalha em *Java* como linguagem padrão.

Na Figura 2 é mostrado o ponto de partida do trabalho, como o objetivo envolve a obtenção e leitura de valores, *MeterReading* foi escolhido como base.

Figura 2 – Perfis equivalentes CIM-*MultiSpeak* mapeados.



Fonte: (EPRI, 2012).

3.2 CONCEITO DE CLASSES E OBJETO

3.2.1 Objetos

Objetos podem ser entendidos por duas visões: conceitual e de implementação. Na primeira visão um objeto representa em um sistema de computação um modelo do mundo real ou que delimite atribuições definidas no sistema. Como exemplo, toma-se um sistema de controle acadêmico, onde são efetuados controles materiais (alunos, professores, entre outros), e outro de entidades abstratas (notas, disciplinas). Dessa linha de raciocínio, os objetos cooperam para executar as tarefas, podendo trocar infor- mações ou controlar outro objeto.

Objetos apresentam três características: identidade (nome ou identifica- ção única que permita endereçamento por outros objetos); atributos (conjunto de informações salvas); e comportamento (conjunto de habilidades que descrevem sua função). Assim, um objeto pode descrever, por exemplo, uma disciplina acadêmica. A identidade sendo “Eletromagnetismo”, com atributos “Carga horária”, “Notas” e “Pré-

requisitos”, e “Média do aluno”, “Média da turma”, “Registrar presença do aluno” como comportamento.

Em visão de implementação, objetos podem ser entendidos como módulos de *software*, logo pode gerar uma saída com base em uma entrada. Da mesma forma, é possível chamar a função do objeto para outro ou ainda enviar mensagens compatíveis com o sistema operacional, portanto um *software* cuja execução depende da colaboração de um conjunto de objetos para realizar determinada função.

Para essa situação a identidade é determinada por um nome dado pelo criador do objeto, por exemplo, uma disciplina, “objMatematica”. No caso do atributo, “nome” poderia ser usado, com valor de “Pedro” ou “Isabela”. É possível criar mais de um atributo, como “rendimento” (valor), “sexo” (M ou F). Por fim, o comportamento, que descreve uma ou mais funções executadas pelo objeto, logo para o caso descrito (objMatematica) poderia existir a função para cálculo da média da turma ou frequência média (STADZISZ, 2022).

3.2.2 Classes

Como os objetos, classes também são compostas em uma visão de conceito e de implementação. Na primeira são definidas pela forma mais abrangente do mundo real, de caráter mais genérico que os objetos. Por exemplo, uma Classe chamada sorvete (chocolate, coco, morango), onde cada componente poderia ser descrito por um objeto, logo um conjunto de entidades pertencentes a um mesmo grupo. Já em sua visão de implementação as classes são compostas por tipos, como inteiro, cadeia de caracteres, real, booleano, entre outros (STADZISZ, 2022).

3.3 NOTAÇÃO UML PARA CLASSES E OBJETOS

3.3.1 Notação para Classes

As classes são apresentadas em UML por meio do retângulo divido em três partes. Detalhes da classe podem ser ocultados em casos em que não se deseje mostrar tais informações, como atributos e métodos, como mostrado na Figura 3.

Figura 3 – Apresentação de classes em UML.

Fonte: (STADZISZ, 2022).

O compartimento superior é reservado para a identificação da classe, mostrando o nome da classe. É recomendado utilizar letras maiúsculas no início do nome e palavras concatenadas, podendo usar a letra C como prefixo para melhor identificação de classe. Observando a o bloco do meio na Figura 3, nota-se sobre o nome da classe os sinais <<>>, estes são denominados estereótipos e são usados para classificar a classe como interface ou controle. Ainda no bloco do meio nota-se "*De PacoteCadastro*", ou de forma geral, "*de nome-do-pacote*", um encapsulamento da classe, portanto o pacote a qual a classe é declarada.

3.3.1.1 Definição de Atributos

Atributos são variáveis pertencentes a classe que pode ser utilizada tanto para escrita quanto para leitura. A notação para um atributo de classe é definida como:

[Visibilidade] Nome-Atributo [Multiplicidade]: [Tipo]=[Valor] [{Propriedades}]

A visibilidade mostra se o atributo pode ser acessado externamente por uma função que não pertença a essa classe, sendo classificada de três formas: visibilidade pública (+), onde é possível a leitura de atributos por funções dentro da classe ou externa a ela; visibilidade privada (-), onde somente funções dentro da classe podem ler o atributo; e visibilidade protegida (#). Em casos na qual a visibilidade não é especificada, é subentendido visibilidade privada.

O nome do atributo é uma cadeia de caracteres que identificam o atributo da classe, com a inicial em minúscula seguida de palavras concatenadas em maiúsculas, podendo ainda ser definida com prefixo “m_”, como m_nivelDeAgua.

Multiplicidade é reservada para casos em que há matrizes ou vetores envolvidos, como vetor de 10 valores (valores [10]) ou matriz de 5 linhas e 10 colunas (matrizDeDados[5,10]).

Tipo, como mencionado anteriormente, indica se os valores são inteiros, caracteres, etc.

Valor inicial, como o nome sugere, indica o valor de partida do atributo, como “valor: int = 5”.

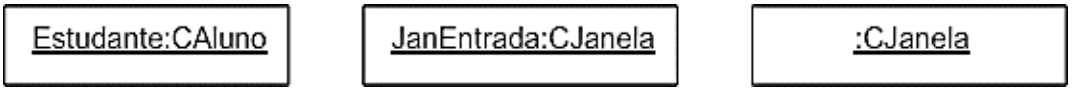
Por fim, as propriedades mostram informações adicionais do atributo por meio de comentários, como se seus valores são contínuos, variáveis, etc., sendo indicadas por chaves.

Para quando se deseja declarar as funções que uma classe possui, são considerados “(Parâmetros)” após o nome do método (Nome-Método), que substitui o nome do atributo, e “[Valor-de-Retorno]”, inserido entre os parâmetros e propriedades do atributo. Os parâmetros são variáveis que recebem valores durante a ativação, definidos pelo nome do parâmetro, seu tipo e seu valor padrão (Nome-do-Parâmetro:Tipo=Valor-Padrão). Já o valor de retorno tem a função de mostrar se o método tem retorno de valores ao final de sua ativação e qual o seu tipo, por exemplo, Valor():int (retorna um valor inteiro).

3.3.2 Notação para Objetos

A Figura 4 mostra a notação UML para um objeto, definida pela identificação do mesmo em seu interior.

Figura 4 – Exemplo de notação para objetos em UML.



Fonte: (STADZISZ, 2022).

A identificação é dada pelo nome do objeto (opcional) acompanhada de dois-pontos (:) e a classe à qual o objeto pertence, sublinhada. Se o nome do objeto for omitido, subentende-se que a referência é um objeto qualquer dentro da classe especificada.

3.4 ESTEREÓTIPOS PARA CLASSES

As classes podem ser formadas por três estereótipos: entidade (engloba dados que simulam o mundo real como aluno, disciplina); controle (controla a execução de processos); e fronteira (responsável pela comunicação com aplicações externas ou atores como teclado, modem, impressora). Para os atores ou aplicações externas é necessário que haja protocolos únicos a cada entidade, de maneira mais organizada deve-se utilizar uma classe para cada ator. Para o controle é recomendado, para um processo mais centralizado e de fácil ajuste, uma classe de controle para cada caso de uso. Demais regras são a definição de controle auxiliares, na qual a classe de controle principal comandaria o fluxo principal do processo, e às classes de controle auxiliares os subprocessos.

Nessa etapa do processo de criação de *softwares* voltados a objetos a maior dificuldade é, portanto, obter as classes que solucionem os casos de uso corretas. Quanto a notação UML para essas classes e objetos, é possível desenvolver o trabalho utilizando conceitos de outras linguagens também focadas em objetos.

3.5 ASSOCIAÇÃO ENTRE CLASSES

Uma associação entre classes nada mais é do que a comunicação entre objetos entre duas classes, podendo ser uni- ou bidirecional. Classes podem se associar a uma ou mais classes, sem limite de associações definido. Sua notação UML é um segmento de reta conectando as diferentes classes. A seta indica o sentido de operação na associação. Para melhor identificação pode-se inserir um nome sobre a seta indicando a natureza da ação.

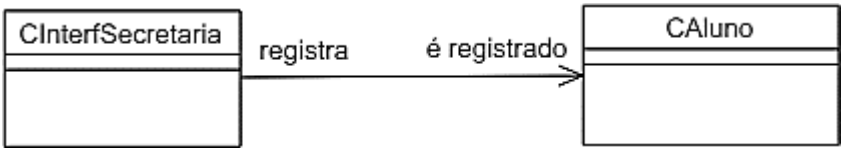
A Figura 5 mostra um exemplo simples de associação entre classes.

Figura 5 – Notação básica para uma associação entre classes.

Fonte: (STADZISZ, 2022).

Da Figura 5, “Registra” é usado como *tag* para melhor clareza, no procedimento unidirecional na qual objetos da classe CInterSecretaria podem se comunicar com os objetos da classe CAluno (por ser unidirecional, os objetos da classe CAluno não conseguem se comunicar com os objetos da classe CInterSecretaria) (STADZISZ, 2022). De forma semelhante, é também possível inserir papéis das classes, uma especifica- ção mais detalhada, que serve para indicar o papel de cada classe, como mostra a Figura 6.

Figura 6 – Notação de exemplo utilizando papéis de classes.



Fonte: (STADZISZ, 2022).

Caso se deseje indicar o número de objetos envolvidos no processo (cardinalidade), basta usar um valor numérico ou simbólico (*) nos extremos do segmento de reta. Se não houver indicação é considerado o valor 1 (somente um objeto de cada classe participa da associação). No Quadro 1 é possível ver alguns exemplos de aplicação de cardinalidade.

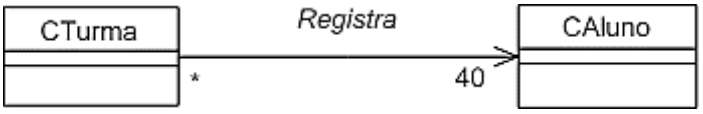
Quadro 1 – Exemplo de valores e significados de cardinalidade.

Notação	Significado	Exemplo
Constante numérica	Indica um número fixo de objetos	5
Faixa de valores	Indica um número variável de objetos dentro da faixa de valores especificada	1..4
Presença ou Ausência	Indica nenhum ou um (ou mais) objetos	0..1
Indefinido	Indica um número qualquer de objetos	*

Fonte: (STADZISZ, 2022).

A Figura 7 ilustra uma associação usando cardinalidade.

Figura 7 – Exemplo do uso de cardinalidade numa associação.



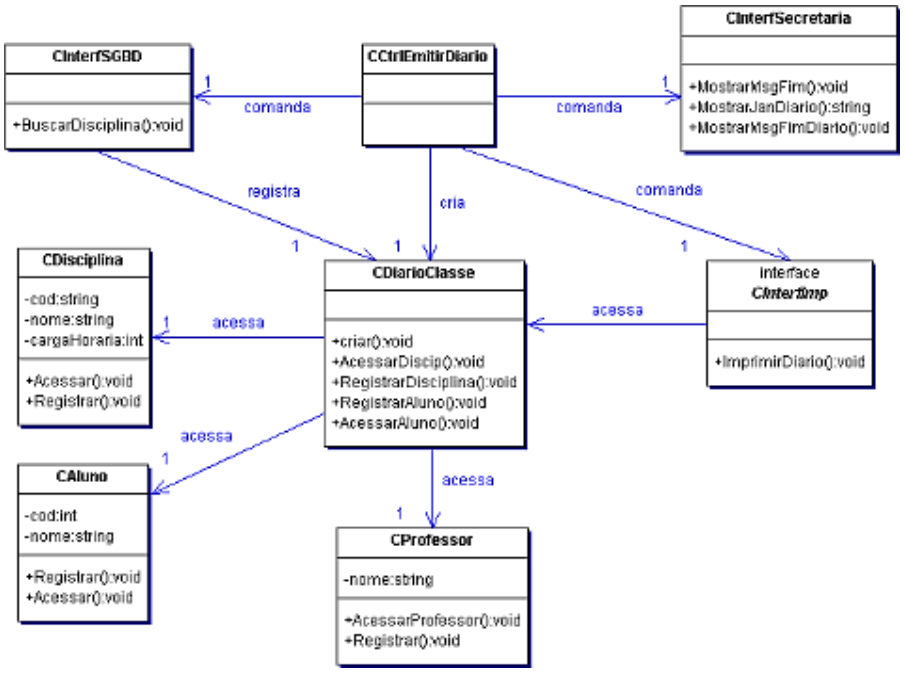
Fonte: (STADZISZ, 2022).

Da Figura 7, a leitura correta do processo é definida como uma associação de um objeto em CTurma com 40 objetos da classe CAluno, enquanto um objeto da classe CAluno se associa com um número indefinido (incluindo zero) de objetos em CTurma, assim é sempre considerado a associação de um objeto da classe de início da leitura com a quantidade especificada na cardinalidade da outra extremidade (STADZISZ, 2022).

3.5.1 Levantamento de Associações

A Figura 8 ilustra um exemplo de levantamento de associações.

Figura 8 – Levantamento de associações.



Fonte: (STADZISZ, 2022).

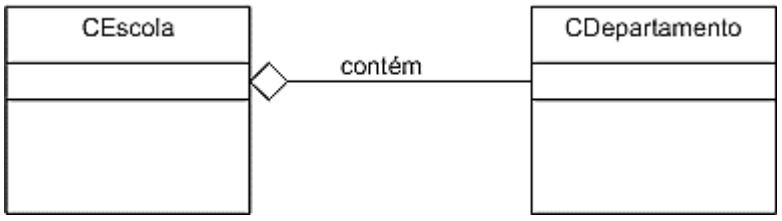
Associações são definidas pela conexão entre duas classes para fins de comunicação entre seus objetos e, para cada par de classes, são verificados se existem mensagens trocadas entre objetos das classes do diagrama de colaboração. Em caso positivo, uma associação uni ou bidirecional é formada de acordo com o sentido das mensagens. Repetindo esse procedimento para todas as classes do sistema, obtém-se uma rede de comunicação entre as classes (STADZISZ, 2022).

3.6 AGREGAÇÃO ENTRE CLASSES

A agregação é uma relação na qual o objeto agregado passa a englobar ou fazer parte do objeto agregador, ressalta-se que a classe não é adicionada dentro de outra apenas objetos dentro de outros objetos. A agregação tem a comunicação unidirecional, do objeto que agrega ao objeto englobado. O agregado não reconhece o objeto que o engloba, portanto não consegue estabelecer uma comunicação em seu sentido. Um objeto pode incluir múltiplos objetos, porém um objeto não pode ser contido por mais de um objeto.

A notação UML para agregações é dada pelo segmento de reta entre as classes dos objetos que agregam aos objetos agregados, com um losango em sua extremidade (no objeto que agrega), conforme ilustrado na Figura 9.

Figura 9 – Notação exemplo para agregações.

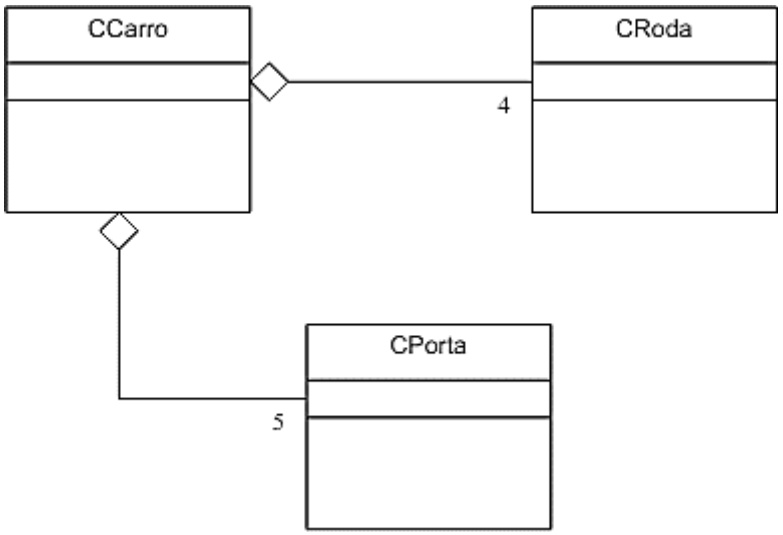


Fonte: (STADZISZ, 2022).

Características como nomes e papéis são opcionais, visto que os nomes não fugiriam do conceito simples de “conter/contém” ou papéis como “contido”. Em termos de cardinalidade, as agregações funcionam como as associações e como objetos têm limite de apenas um objeto que os agrega, agregadores terão sempre a cardinalidade 1.

A Figura 10 ilustra a agregação com cardinalidade.

Figura 10 – Exemplo de agregações com cardinalidade.



Fonte: (STADZISZ, 2022).

Da Figura 10, um objeto da classe CCarro contém 5 objetos da classe CPorta e 4 objetos da classe CRoda.

3.6.1 Tipos de Agregação

Agregações podem ser divididas por composição (onde é criado ou alterado a posição de um objeto dentro de outro objeto, visualmente indicado por um retângulo preenchido), por associação (a alocação do objeto é dada de forma estática ou dinâmica, fora ou dentro, respectivamente, do objeto que agrega). Os dois métodos descritos se diferenciam dependendo se o número de objetos e fixo (composição) ou variável (associação). A agregação por associação é dada pelo retângulo não preenchido.

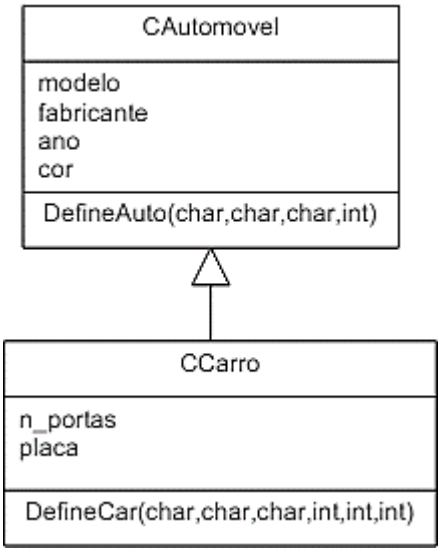
3.7 GENERALIZAÇÃO/ESPECIALIZAÇÃO DE CLASSES

No envolvimento de duas classes, uma será considerada superclasse, sendo tratada como classe base, enquanto a outra será tratada como classe derivada ou

subclasse. Se a leitura for no sentido da classe base para a classe derivada o processo é definido como uma especialização, ou seja, a classe derivada se torna uma classe especial da classe base, caso contrário a relação se torna uma generalização, onde a classe base passa a ser um caso geral da classe derivada. De forma resumida, o conceito de generalização ou especialização sempre se dá pela herança da classe base para a classe derivada. A classe derivada adota todos os atributos e métodos da classe base, além de atributos e métodos extras além dos herdados. A principal diferença entre uma agregação e herança é, portanto, que o objeto incorporado herda tudo que a classe generalizadora tem, enquanto na agregação uma classe agregada usa outra para existir, mas podendo existir sem ela. É descrita pelo segmento de reta com o triângulo em uma de suas extremidades.

A Figura 11 demonstra um exemplo de herança entre CAutomovel e CCarro. Nesse caso, há uma generalização entre as classes, de forma que CCarro mantém seus atributos n_portas e placa mais os atributos da classe CAutomovel.

Figura 11 – Exemplo de herança.

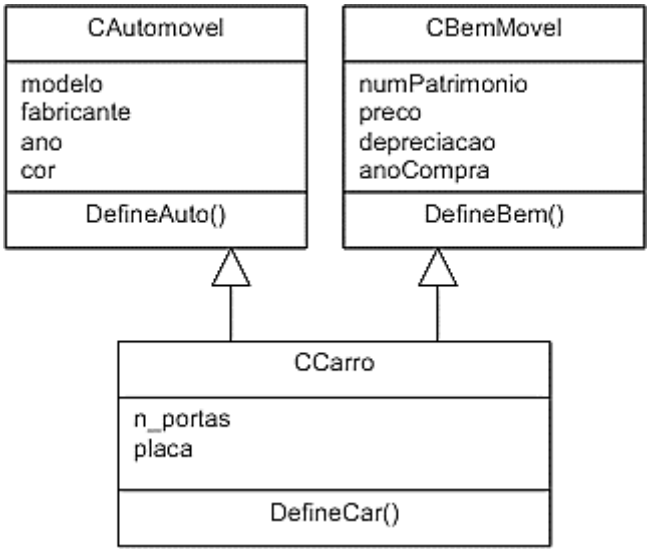


Fonte: (Stadzisz, 2022).

Para casos de herança o uso de nomes é opcional, já que a natureza do relacionamento é apenas uma, logo é dispensado também o uso de cardinalidade devido a relação ser dada por uma única classe para outra classe.

Caso exista mais de uma classe base por classe derivada, é considerado uma herança múltipla. As propriedades de uma herança são mantidas para esses tipos, com a diferença de que a classe derivada mantém os atributos e métodos de todas as classes bases. A diferença passa a ser que, por apresentar mais de uma classe base, o conceito de generalização diminui. Para a Figura 12, CAutomovel representa uma generalização parcial juntamente com CBemMovel, onde CCarro passa a ser a especialização de CAutomovel e CBemMovel (STADZISZ, 2022).

Figura 12 – Exemplo de herança múltipla.



Fonte: (STADZISZ, 2022).

3.8 CONVENÇÃO DOS DIAGRAMAS UML

Em alguns casos surge a necessidade de abrir sugestões no momento de adição ao *MultiSpeak* necessárias para fazer o perfil mais consistente com o CIM. Na maioria das vezes, essa adição acaba sendo uma associação ou ainda uma classe abstrata, podendo ser a classe de suporte parental em qualquer perfil. A classe *AuxiliaryAgreementConfig*, por exemplo, mantém diversos *AuxiliaryAgreements*, mas como *MultiSpeak* não apresenta essa classe, para acomodar essa classe e suas derivações e relações, é adicionado uma classe abstrata, que serve para sugerir como uma classe equivalente pode ser criada. Isso é repassado para o Comitê Técnico responsável pelo *MultiSpeak*, podendo então ser atualizado em futuras edições.

3.9 CONVENÇÃO DE TABELAS PARA IDENTIFICAR CLASSES E ATRIBUTOS

No Quadro 2 é mostrado um exemplo de convenção para identificar os atributos de uma classe. As três colunas mostram as classes e atributos para CIM, *MultiSpeak* e comentários, respectivamente, que podem ser adicionados para facilitar o procedi- mento de mapeamento.

Quadro 2 – Exemplo de convenção CIM-*MultiSpeak* de uma classe.

CIM	MultiSpeak	Comments
MeterAsset .mRID .category	Customer .objectID	

Fonte: (EPRI, 2012).

No caso da coluna do modelo CIM, tomando a classe de exemplo “*MeterAsset*”, são mostrados os atributos “*mRID*” e “*category*”, ambos denotados pelo ponto. Para *MultiSpeak*, “*objectID*” é o atributo referente a classe “*Customer*”.

3.10 METER READINGS

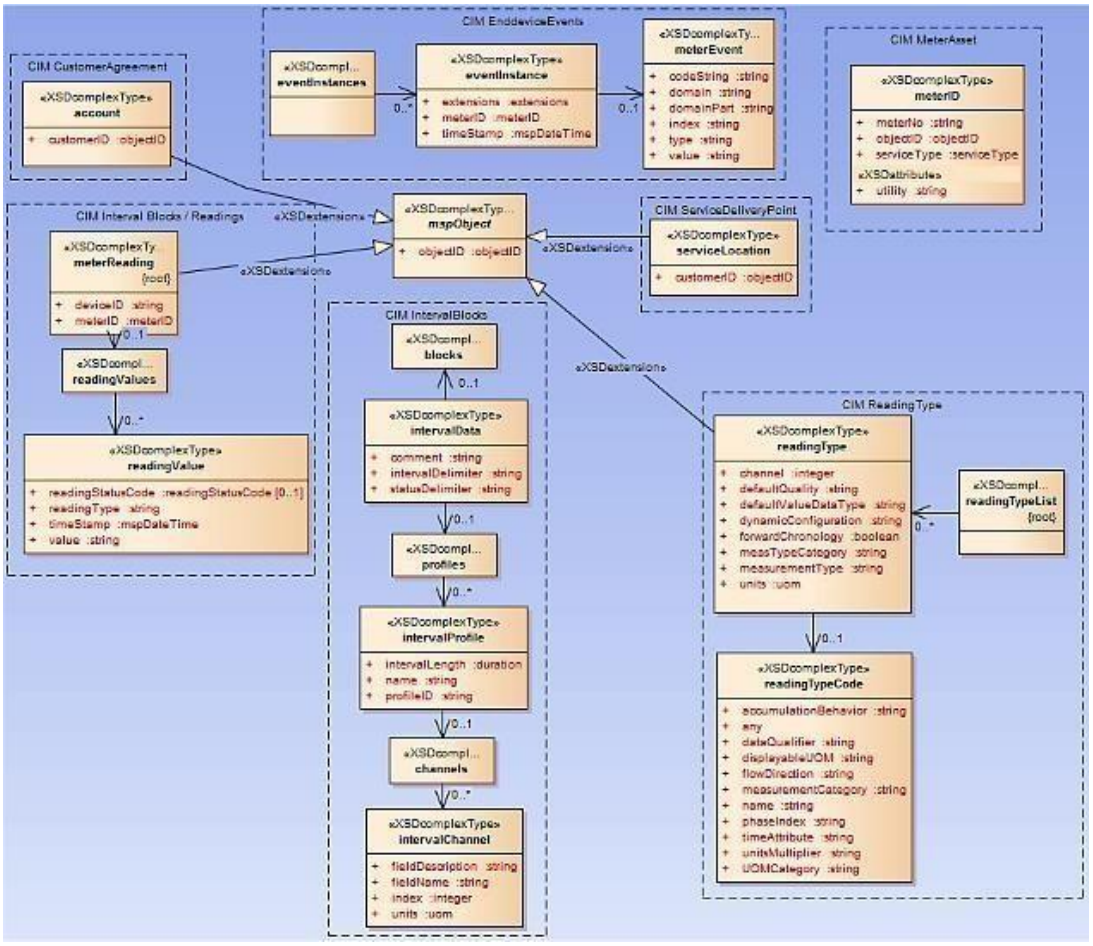
O perfil *MeterReadings* pertence ao modelo CIM, e nele são contidas duas estruturas principais, *MeterReading* e *ReadingType*. Dentro do mesmo há ainda *EndDeviceEvents*, *CustomerAgreement* (identificador do cliente), *MeterAsset* (identificador da medida), *IntervalBlocks*, *Readings* e o *ServiceDeliveryPoint*. A Figura 13 mostra o equivalente *MultiSpeak* do perfil *MeterReadings* (EPRI, 2012).

O Quadro 3 mostra as correlações existentes para a classe *MeterReadings*, composta pelas classes *ReadingType* e *Readings*.

O Quadro 4 mostra as transformações necessárias entre os dois padrões.

No Quadro 5 os *gaps* existentes entre os dois modelos.

Figura 13 – Perfil equivalente CIM-*MultiSpeak* para *MeterReadings*.



Fonte: (EPRI, 2012).

Quadro 3 – Correlação CIM-*MultiSpeak* para a classe *MeterReadings*.

CIM	MultiSpeak	Comments
ReadingType .mRID .channelNumber .defaultQuality .defaultValueDataType .dynamicConfiguration .multiplier .name .unit	readingType .objectID .channel .defaultQuality .defaultValueDataType .dynamicConfiguration .unitsMultiplier .name .units	
Readings .timestamp .value	readingValues .timestamp .value	

Fonte: (EPRI, 2012).

Quadro 4 – Transformação CIM-*MultiSpeak* para a classe *MeterReadings*.

CIM	MultiSpeak	Comments
CustomerAgreement .mRID	customerID	A customerID in MultiSpeak is of type objectID, which can be used to identify a given customer
intervalLength	timeAttribute	
kind	measTypeCategory	
reverseChronology	forwardChronology	The intent is to indicate chronology, one is simply the inverse of the other
ReadingQualties .quality	readingStatusCode .codeCateogry .codeIndex .originatingSystemID	These correspond to IEC 61968-9 Annex C
IntervalBlocks	blocks	

Fonte: (EPRI, 2012).

Quadro 5 – *Gaps* da classe *MeterReadings* na transformação CIM-*MultiSpeak*.

CIM	MultiSpeak	Comments
	readingTypeList	MultiSpeak has the capability through the list class, to associate many readingTypes.
Pending .multiplyBeforeAdd .offset .scalarDenominator .scalarFloat .scalarNumerator		
	intervalData .comment .intervalDelimiter .statusDelimiter	
	profiles	
	intervalProfile .intervalLength .name .profileID	
	channels	
	intervalChannel .fieldDescription .fieldName .index .units	

Fonte: (EPRI, 2012).

3.11 CRIAÇÃO DE INTERFACES IDC EM *SMART GRIDS*

Partindo do conceito de mover o SCADA para a nuvem com a possibilidade de menor custo em montagem e equipe técnica de manutenção de *software* e *hardware*, além da utilização da otimização da energia residencial com HEMS, é possível estabelecer um DMS avançado, capaz de integrar em tempo real, ou quase tempo real, medições avançadas, combinando tecnologia da informação (TI) e *OT* pelo uso de dados para um grande al- cance de equipamentos a fim de reduzir custos de operação e aumentar a produtivi- dade e confiança de funcionamento. O padrão CIM providencia o modelo de dados para o DMS, o que facilita acesso dos centros de aplicações ao IEC 61850 (por exem-plo) e demais dados pelo CIM.

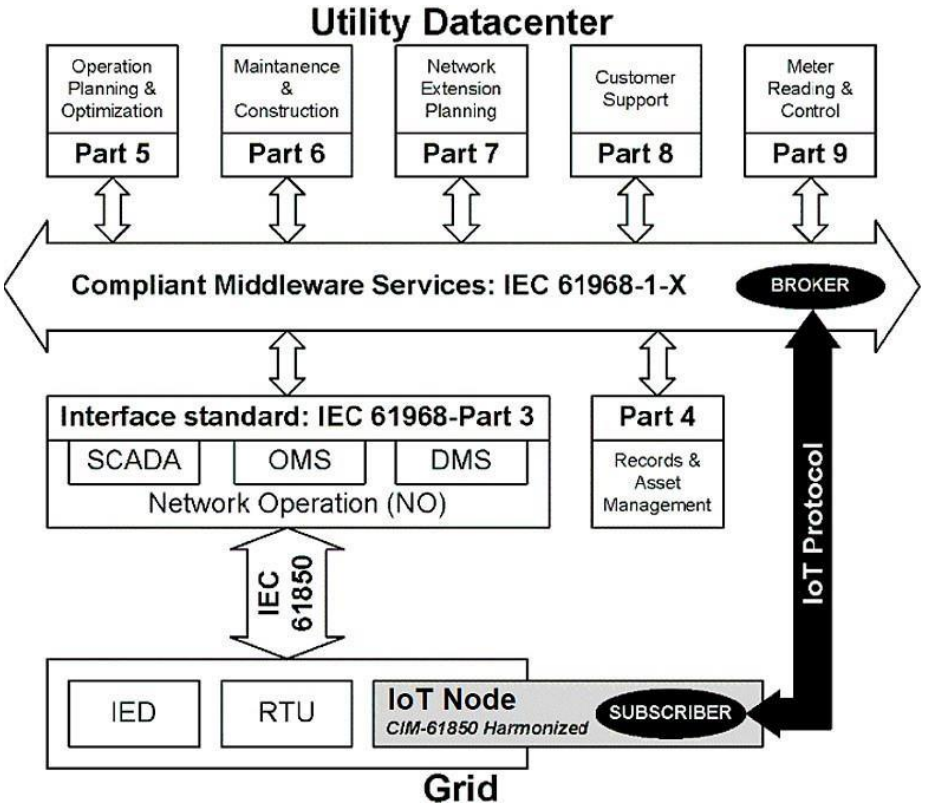
A Figura 14 mostra o modelo do nó implantado no ambiente de operações da rede, o ambiente integrado é suportado pelos padrões IEC 61850, 61970/61968 o que torna flexibilidade quando implementando aplicações *smart grid*.

O ambiente com SCADA, OMS, e DMS integrado como uma rede em um serviço de operação na rede, por exemplo, um serviço de web que troca dados informação usando um barramento de serviço coorporativo, do Inglês, *Enterprise Service Bus* (ESB), depende da harmonização entre CIM e IEC 61850. ESB troca mensagens usando esquemáticos XML através de interfaces padronizadas bem definidas, logo a harmonização com CIM permitiria a troca direta dessas informações com interfaces padronizadas via protocolos IdC (Leite & Kezunovic, 2023).

3.11.1 Protocolo MQTT e Serviço de *Broker*

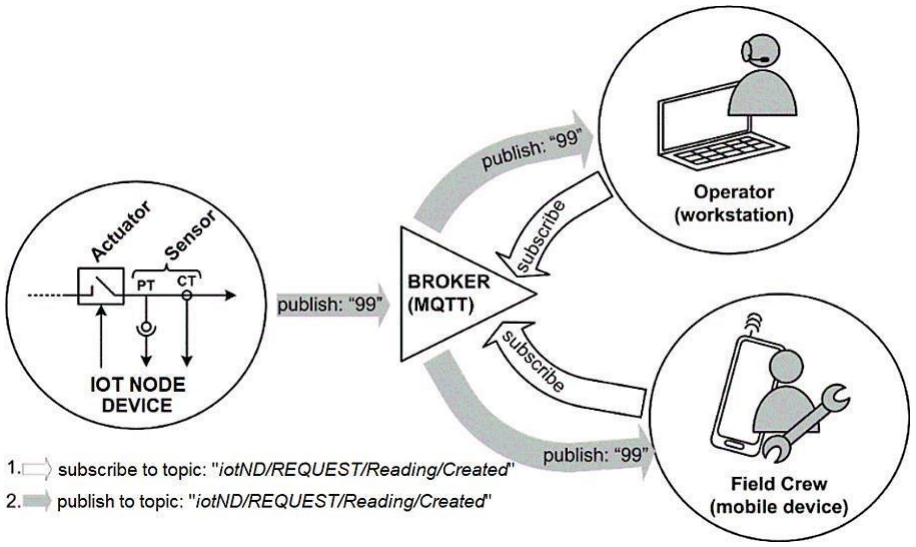
O protocolo *Message Queuing Telemetry Transport* (MQTT) é baseado em *broker* (ou corretor), também classificado como servidor, podendo ser hospedado localmente ou por um serviço baseado em nuvem. Nesse sistema um publicador ou editor posta uma mensagem ao corretor que então passa a mensagem ao cliente inscrito ao tópico especificado pelo editor. O sistema também permite que o inscrito seja linkado em subtópicos específicos. Em uma aplicação real com diversas redes diferentes, é trabalho do DMS reconhecer o identificador de recursos mestre (do Inglês *mRID*) ou o nome do nó IdC na qual o dispositivo está enviando mensagens.

Figura 14 – Dispositivo IdC harmonizado no sistema de distribuição.



Fonte: (Leite & Kezunovic, 2023)

Figura 15 – Sistema de inscrição e publicação no protocolo MQTT.



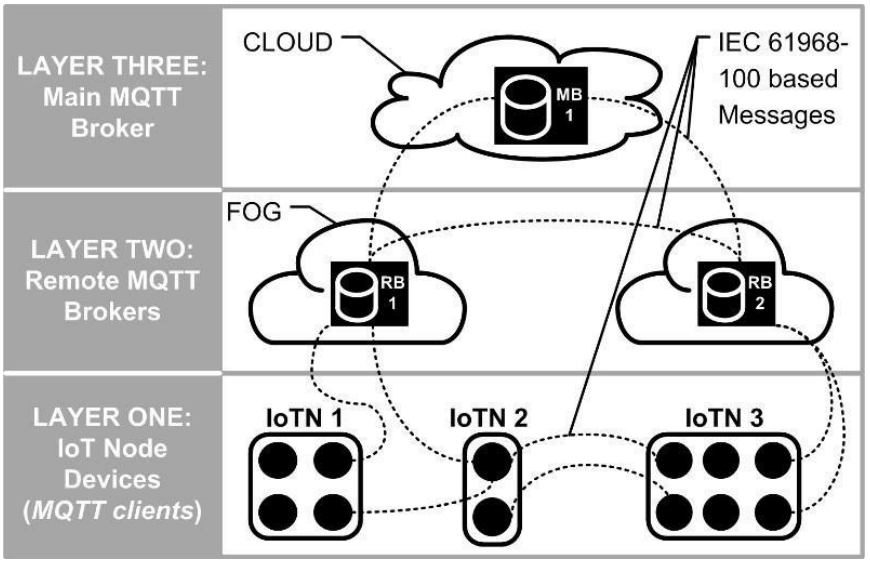
Fonte: (Leite & Kezunovic, 2023).

Em uma situação em que uma mensagem seja enviada pelo editor onde não há inscritos naquele tópico, o editor pode optar pela retenção da mensagem, caso

contrário o corretor apaga a mensagem, assim qualquer inscrição feita no tópico na situação que a mensagem foi retida o inscrito tem acesso ao conteúdo.

O sistema de nuvem resolve muitos dos problemas envolvendo IdC, como localização, mobilidade, e distribuição, porém falha em suportar redes onipresentes devido a delays, conectividade e banda de comunicação limitada. Dividindo o procedimento em camadas é introduzido um nível entre nuvem e dispositivos IdC. A Figura 16 ilustra as três camadas: a primeira hospeda os dispositivos IdC próximos à rede de elétrica, ou processos de energia; a segunda os corretores remotos que dão assistência ao corretor principal na computação em nuvem; por fim, a terceira camada que abriga o corretor MQTT principal.

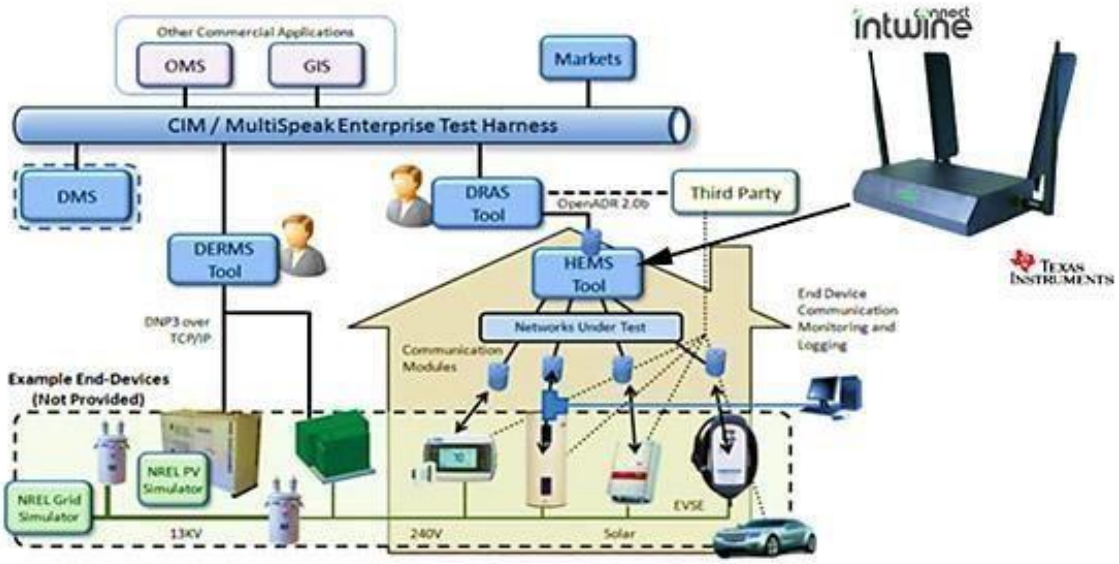
Figura 16 – Estrutura de dispositivos IdC em um sistema de multicamadas no modelo de mensagens IEC 61968.



Fonte: (Leite & Kezunovic, 2023).

A arquitetura multicamadas da IdC através do protocolo MQTT, possibilita dispositivos de borda, próximo ao processo da eletricidade, trocar informações padronizadas, ou seja, mensagens CIM-IEC 61850, com aplicações de software de central de operação das concessionárias de distribuição. Da mesma forma os dispositivos dos clientes de energia podem trocar suas informações tornando o ESB compatível com CIM e Multspeak. Na Figura 16, é apresentado um exemplo de sistema com ESB harmonizado que permite a integração DMS e HEMS.

Figura 16 – ESB harmonizado CIM – *Multispeak.*



Fonte: ????

Essa flexibilidade adicional é garantida usando modelos harmonizados CIM-Multispeak, conforme propostos e desenvolvidos neste trabalho de graduação, permitindo às aplicações de *software* do DMS obterem informações diretamente do HEMS.

4 MODELOS HARMONIZADOS EM LINGUAGEM DE PROGRAMAÇÃO

Os resultados apresentados neste capítulo foram obtidos com o auxílio do software *Enterprise Architect* e do site *replit,* para suporte dn linguagem C#, em conjunto com o software *Visual Studio Code*. Para que não haja erros no código, é essencial primeiro estabelecer as bibliotecas (pacotes) que contenham cada classe ou elemento apresentado utilizado no programa. Na Figura 17 são apresentados os pacotes necessários para desenvolver o perfil de compatibilidade CIM-*MultiSpeak*.

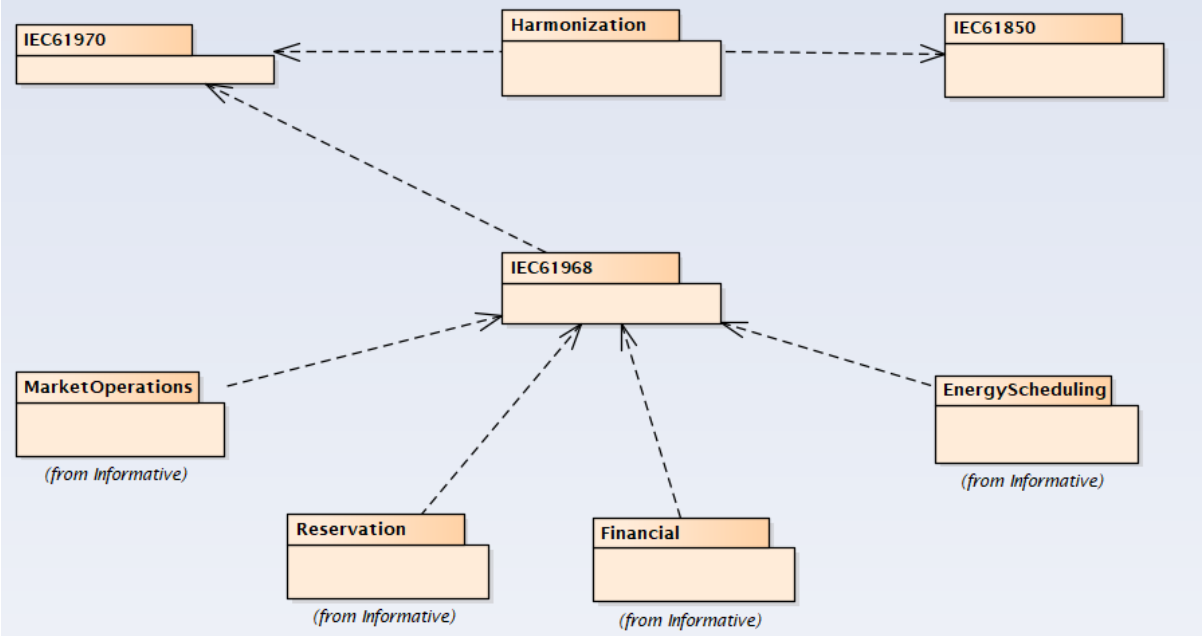
Figura 17 – Pacote para compatibilidade das classes para a harmonização.

```
//IEC 61970, IEC 61968 e IEC 61850
using TC57CIM.IEC61968.AssetModels;
using TC57CIM.IEC61968.Assets;
using TC57CIM.IEC61970.Meas;
using TC57CIM.IEC61970.Domain;
using TC57CIM.IEC61968.Common;
using TC57CIM.IEC61968.Metering;
using TC57CIM.IEC61970.Core;
using TC57CIM.IEC61968.Customers;
using TC57CIM.IEC61850.Collections;
using TC57CIM.IEC61970.LoadModel;
using TC57CIM.IEC61970.Topology;
using TC57CIM.Informative.EnergyScheduling;
using TC57CIM.IEC61968.PaymentMetering;
using TC57CIM.IEC61968.Informative.InfAssets;
using TC57CIM.IEC61968.Informative.InfPaymentMetering;
using TC57CIM.IEC61968.Informative.InfCustomers;
using TC57CIM.IEC61968.Informative.InfTypeAsset;
using TC57CIM.IEC61968.Informative.InfAssetModels;
using TC57CIM.Harmonization.ISO15926;
using TC57CIM.Harmonization;
using TC57CIM.IEC61968.Informative.InfWork;
using TC57CIM.IEC61968.Informative.InfERPSupport;
using TC57CIM.IEC61968.Informative.InfCommon;
using TC57CIM.IEC61968.Informative.InfLocations;
using TC57CIM.IEC61968.Informative.InfOperations;
using TC57CIM.Informative.MarketOperations;
using TC57CIM.IEC61968.WiresExt;
using TC57CIM.IEC61968.Work;
using TC57CIM.IEC61970.SCADA;
using TC57CIM.Harmonization.SCADA;
using TC57CIM.IEC61850.IEC61850Data;
```

Fonte: Elaborado pelo autor.

Uma maneira mais simples de visualizar e entender como os pacotes se relacionam entre si é ilustrada pela Figura 18. Cada linha da linha de código na Figura 17 representa um pacote e suas ramificações, onde estão contidas as classes requeridas para a harmonização, por exemplo, a linha “TC57CIM.IEC61968.Work” indica que o pacote *Work* (necessário para o funcionamento da classe *Work*) está contido no pacote IEC61968, que, por sua vez, está contido em TC57CIM (Enterprise Architect®).

Figura 18 – Diagrama UML principal.



Legenda: os segmentos de reta tracejados indicam dependência, ou seja, o elemento fonte depende do alvo e pode ser afetado por alterações nele.

Fonte: Elaborado pelo autor.

Junto as definições dos pacotes são inseridas as principais classes, ou seja, as classes compatíveis com os padrões CIM e *MultiSpeak*.

Figura 19 – Perfil para o sistema de gerenciamento de energia residencial.

```
namespace CIM_MultiSpeak
{
    0 references
    class HEMS
    {
        0 references
        class AuxiliaryAgreementConfig ···
        0 references
        class CustomerConfig ···
        0 references
        class CustomerMeterDataSet ···
        0 references
        class EndDeviceAssets ···
        0 references
        class EndDeviceEvents(Updated) ···
        0 references
        class MeterAssetReading ···
        0 references
        class MeterReadings ···
        0 references
        class MeterServiceRequests ···
        0 references
        class MeterSystemEvents ···
        0 references
        class PricingStructureConfig ···
        0 references
        class ReceiptRecord ···
        0 references
        class SDPLocationConfig ···
        0 references
        class SupplierConfig ···
        0 references
        class TransactionRecord ···
    }
}
```

Fonte: Elaborado pelo autor.

Cada elemento é importante para a criação da classe HEMS, contida no *namespace* CIM_*MultiSpeak.* Ao expandir os elementos, por exemplo, *CustomerConfig*, Figura 20, nota-se que HEMS tem como atributos outros tipos padronizados.

Figura 20 – Exemplo de atributos da classe HEMS.

```
class CustomerConfig

    {
0 references | 0 references
public TC57CIM.IEC61968.Customers.CustomerKind CustomerKind;public TC57CIM.IEC61968.Customers.Customer Customer;
0 references
public TC57CIM.IEC61968.Common.ElectronicAddress ElectronicAdresses;

}
```

Fonte: Elaborado pelo autor.

Para identificar quais atributos devem formar HEMS, é necessário obter para cada uma das classes apresentadas na Figura 2 as equivalências entre os modelos (CIM e *MultiSpeak*), a exemplo de *CustomerConfig,* estas são *CustomerKind e ElectronicAddress.* No Quadro 5 as transformações que compõem essa classe.

Quadro 5 – Transformação CIM-*MultiSpeak* para a classe *CustomerConfig.*

CIM	MultiSpeak
Customer .specialNeed .vip	Customer .specialNeeds
Customer.kind	revenueClass
ElectronicAdresses .lan .name .password .radio .userID .web	otherContactInformation otherContactItem .details .infoType

Fonte: (EPRI, 2012).

Ao encontrar as classes necessárias é preciso inseri-las dentro de HEMS, isso é feito escrevendo o caminho na qual se encontram na biblioteca ou pacote geral (ver

Figura 17) e inserindo *public* no início da linha de código. Ao final da linha do código é dado um nome a esse novo atributo, “TC57CIM.IEC61968.*Customers.Customer Customer*”, por exemplo, que segue o nome da classe original.

Usando como exemplo de um tipo compatível com ambos os modelos o *AuxiliaryAgreement*, Figura 21, é inserido um termo dentro do código fonte classificado como “*MultiSpeak*”.

Figura 21 – Exemplo de um tipo compatível entre CIM e *MultiSpeak*.

```
//////////////////AuxiliaryAgreement
namespace TC57CIM.IEC61968.PaymentMetering
{
    4 references
    public class AuxiliaryAgreement : Agreement
    {

//////MULTISPEAK
        0 references
        public TC57CIM.IEC61970.Domain.String objectID;
        0 references
        public TC57CIM.IEC61970.Domain.String Description;
        0 references
        public TC57CIM.IEC61970.Domain.String Name;
        0 references
        public TC57CIM.IEC61970.Domain.String objectName;
        0 references
        public Status statusCode;
```

Fonte: Elaborado pelo autor.

O termo inserido fornece o link para que haja a harmonização entre os modelos. Como *MultiSpeak* apresenta mapeamento para *AuxiliaryAgreementConfig* (que contém *AuxiliaryAgreement*), mas não *AuxiliaryAgreement*, este processo se torna necessário. Ele é desenvolvido obtendo o termo mais comum e básico no programa geral, ou seja, procurar de onde a classe de escolha está herdando e, para o *AuxiliaryAgreement*, esta classe é o *Agreement*. O processo é repetido mais duas vezes e, como ilustra a Figura 22, a herança *Document* é encontrada dentro de *Agreement*, que, por sua vez, herda de *IdentifiedObject* contido no pacote *Core*.

Figura 22 – Herança para a classe *Agreement*.

```
/////////////////Agreement
namespace TC57CIM.IEC61968.Common
{

    4 references
    public class Agreement : Document
    {

    }
}

/////////////////Document
namespace TC57CIM.IEC61968.Common
{

    57 references
    public class Document : IdentifiedObject
    {
```

Fonte: Elaborado pelo autor.

A Figura 23 representa o código da classe *IdentifiedObject*, uma das classes mais importantes para o processo de harmonização, devido ao seu papel de estabelecer uma nomeação comum aos atributos de todas as classes que precisam dos atributos de nomeação.

Figura 23 – Classe *IdentifiedObject*.

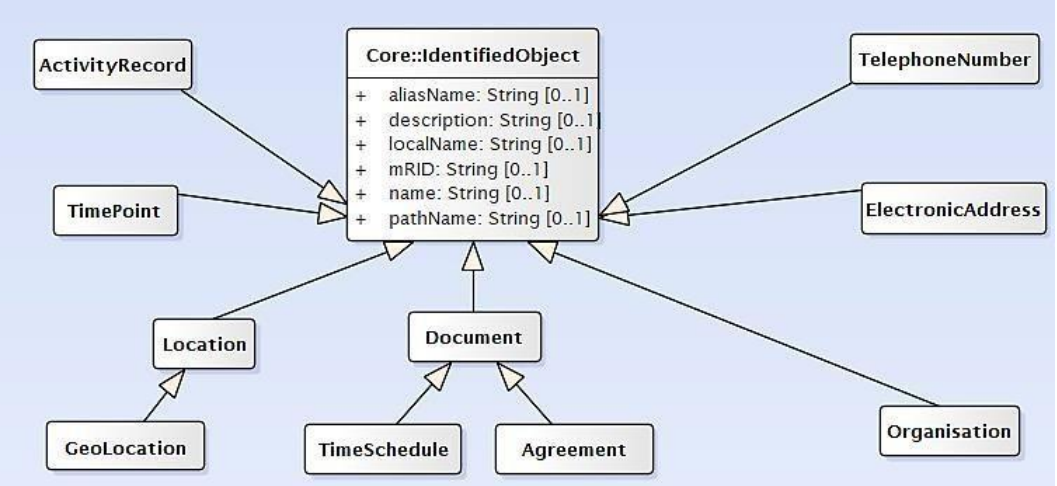
```
////////////////////////// IdentifiedObject
namespace TC57CIM.IEC61970.Core
{

    97 references
    public class IdentifiedObject
    {

        0 references
        public TC57CIM.IEC61970.Domain.String aliasName;
        0 references
        public TC57CIM.IEC61970.Domain.String description;
        0 references
        public TC57CIM.IEC61970.Domain.String localName;
        0 references
        public TC57CIM.IEC61970.Domain.String mRID;
        0 references
        public TC57CIM.IEC61970.Domain.String name;
        0 references
        public TC57CIM.IEC61970.Domain.String pathName;
        0 references
        public TC57CIM.IEC61970.Core.ModelingAuthoritySet ModelingAuthoritySet;
        0 references
        public IdentifiedObject()
        {

        }

        0 references
        ~IdentifiedObject()
        {

        }

    }//end IdentifiedObject

}//end namespace Core
```

Fonte: Elaborado pelo autor.

Devido a importância de *IdentifiedObject* para a harmonização, são necessários que seus atributos estejam dentro dos moldes aceitos por *MultiSpeak*, portanto suas *strings* devem ser transformadas (*mRID* em *objectID*, *description* em *Description*, *aliasName* em *objectName*, entre outros), representado na Figura 21.

Na Figura 24 a visualização do processo de herança, tomando o pacote *Comum* que contém as classes *Document* e *Agreement*.

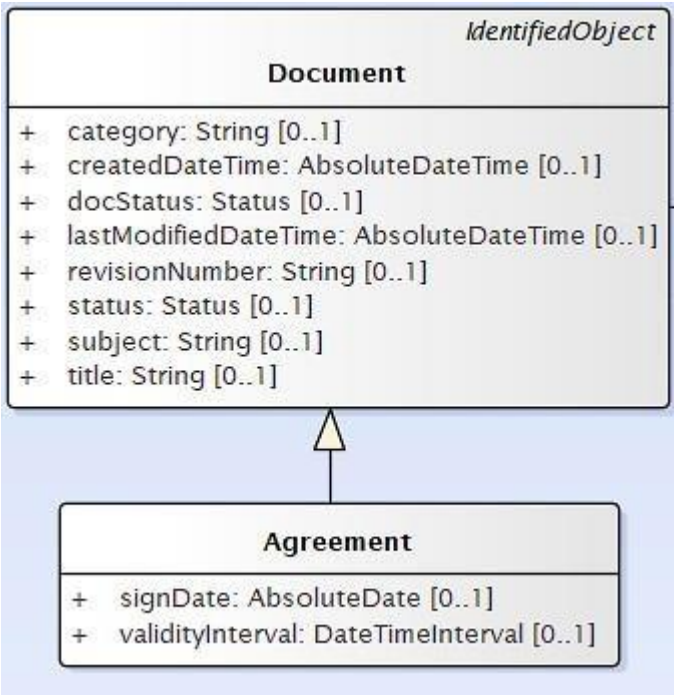
Figura 24 – Diagrama UML de herança no pacote *Comum.*



Legenda: os segmentos de reta indicam o processo de generalização.

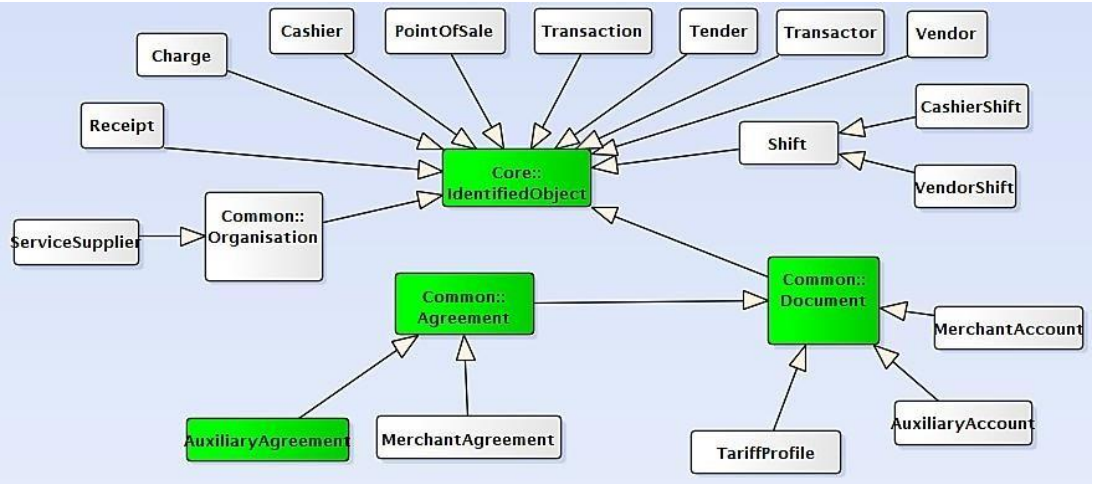
Fonte: Elaborado pelo autor.

Figura 25 – Visualização detalhada do processo entre *Agreement* e *Document.*



Fonte: Elaborado pelo autor.

Figura 26 – Visualização geral do processo exemplo pela herança do pacote *PaymentMetering.*



Legenda: em verde as classes escolhidas como exemplo para descrever o procedimento para criação dos atributos necessários a HEMS.

Fonte: Elaborado pelo autor.

5 CONCLUSÃO

Este trabalho de graduação foi realizado com objetivo principal de desenvolver uma interface de compatibilidade entre os modelos CIM e MultiSpeak para padronizar as trocas de informações entre o DMS e o HEMS. O software Enterprise Architect juntamente com um software de compilação de linguagem de programação, em particular o C#, foram utilizados no desenvolvimento do modelo de compatibilidade.

O perfil criado para modelar o HEMS tem classes de diferentes pacotes padronizados pela TC57 que permitiu fazer as adaptações necessárias para chegar ao modelo compatível CIM e Multispeak. O desenvolvimento deste modelo harmonizado somente foi possível com o entendimento da UML, ou seja, dos diagramas de classes e suas relações do CIM observadas usando o software Enterprise Architect. O modelo desenvolvido segue as recomendações do EPRI, porém seus objetos e funcionalidades não foram testados.

Para trabalhos futuros, recomenda-se uma interface gráfica do HEMS deve ser elaborada para possibilitar a construção dos objetos harmonizados CIM-Multspeak e para verificar as trocas de mensagens entre o DMS e HEMS através da arquitetura IdC multicamadas.

REFERÊNCIAS

EPRI, Dezembro, 2012, “CIM-MULTISPEAK HARMONIZATION 2ND EDITION”. Disponível em: https://www.epri.com/research/products/1026585. Acesso em 24 de março de 2024

McNaughton, G. A., G. Robinson, G. R. Gray “MultiSpeak and IEC 61968 CIM: Moving Towards Interoperability” in Grid-Interop Forum 2008

Simmins, maio, 2011, “The impact of PAP 8 on the Common Information Model (CIM)”. Disponível em: https://ieeexplore.ieee.org/document/5772503/authors#au- thors. Acesso em 13 de abril de 2024.

Leite, J. B., M. Kezunovic. “Integrating IoT Devices with Distribution Energy Management System by Harmonizing Their Logical Models Using IEC Standards 61970/61968 and 61850.” In: 2023 IEEE Power & Energy Society General Meeting (PESGM), 2023, Orlando. 2023 IEEE Power & Energy Society General Meeting (PESGM). Piscataway: IEEE, 2023. p. 1.

Kristin Swenson, “Reliability First, Order 2222 & Order 2222-A”, Midwest Independent System Operator., April 19, 2021. Available: https://rfirst.org/ProgramAreas/ESP/RAPA%20Library/2021-0419%20MISO%20Distributed%20Energy%20Resources.pdf

MultiSpeak, “What is MultiSpeak?”. Disponível em: https://www.multispeak.org/what-is-multispeak/. Acesso em 25 de março de 2024.

NIST, January, 2010, “National Institute of Standards and Technology, NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0”. Disponível em: https://www.nist.gov/publications/nist-framework-and-roadmap-smart-grid-interoperability-standards-release-10. Acesso em 26 de março de 2024.

P. C. STADZISZ, “Projeto de Software usando a UML”, CEFET-PR, 2022

Power Europe, Maio, 2023, “DSO, TSO, and Power Generator Utility Webinar & Digital Presentation Schedule”. Disponível em: https://www.power-europe.com/2021-webinar-schedule.html. Acesso em: 14 de abril de 2024.

Sidney L. Mannheim, Jeremy Malekos “Docket No. ER06-615-000 Public Version of Annual Report Evaluating Demand Response Participation in the CAISO for 2019”, California Independent System Operator Corporation, January 16, 2020.

W. MULLENMASTER, T. Nielsen, M. Highfill, S. Sternfeld. “RESTful Microservice Architec- ture for Electric Utility Distribution Systems,” in CIGRE US National Committee 2022 Grid of the Future Symposium, Pariz, France, 2022