**ESCOLA POLITÉCNICA
UNIVERSIDADE DE SÃO PAULO**

CARLOS AUGUSTO SIQUEIRA DA CUNHA JÚNIOR

**COMMON INFORMATION MODEL / EXTENSIBLE MARKUP LANGUAGE
(CIM/XML) NA TROCA DE INFORMAÇÕES ENTRE CENTROS DE CONTROLE
DE ENERGIA ELÉTRICA**

São Paulo
2005

CARLOS AUGUSTO SIQUEIRA DA CUNHA JÚNIOR

**COMMON INFORMATION MODEL / EXTENSIBLE MARKUP LANGUAGE
(CIM/XML) NA TROCA DE INFORMAÇÕES ENTRE CENTROS DE CONTROLE
DE ENERGIA ELÉTRICA**

Dissertação apresentada à Escola Politécnica
da Universidade de São Paulo para obtenção
do título de Mestre em Engenharia.

Área de Concentração:
Energia e Automação de Sistemas Elétricos.

Orientador:
Prof. Dr. José Antonio Jardim.

São Paulo
2005

AGRADECIMENTOS

Ao orientador Prof. Dr. José Antonio Jardini pelo apoio e confiança dispensada, sem o qual este trabalho não teria se concretizado.

Ao amigo e grande incentivador Prof. Dr. Luiz Carlos Magrini pelos ensinamentos, conselhos, estímulos e paciência dedicada durante todo o processo de mestrado.

À amiga Patrícia Rodrigues, pela inestimável ajuda e dedicação empregada na carga das informações do Banco de Dados.

À Consist, representante no Brasil da Software AG, pela confiança depositada na concessão de licença trabalhista para conclusão desta pesquisa.

Aos amigos e colegas de trabalho pelo apoio, carinho e grande torcida.

À todos que direta e indiretamente colaboraram na concretização deste trabalho.

RESUMO

CUNHA JR., C. A. S. **Common Information Model/Extensible Markup Language (CIM/XML) na troca de informações entre Centros de Controle de Energia Elétrica.** 2005. 232 p. Dissertação Mestrado – Escola Politécnica, Universidade de São Paulo, São Paulo.

Esta dissertação analisa a utilização do modelo de dados padronizado pela norma IEC 61970 (CIM), tida como uma ferramenta para troca de informações cadastrais e operacionais entre empresas de energia elétrica, com sistemas computacionais de diferentes fabricantes. O propósito deste padrão é criar um mecanismo para troca de informações baseado em XML, denominado de CIM/XML – um formato, especificamente, utilizado pelos Centros de Controle de Energia Elétrica para troca de dados. Os modelos de dados do padrão IEC 61970 são apresentados, bem como a avaliação do CIM/XML como uma ferramenta de interoperabilidade de dados entre empresas de energia que apresentam bases de dados de diferentes modelagem e implementação. Um dos méritos deste modelo, além de fazer uso de uma tecnologia aberta (XML) disponível em qualquer tipo de computador, e que possibilita armazenar e transferir não só dados cadastrais de equipamentos, mas também dados de topologia da rede, curvas de carga, programação de geração, saídas programadas de equipamentos, bem como medições de sistemas SCADA, indicações de estado e alarmes. Possibilita ainda armazenamento de resultados de simulação, tais como resultados do programa de fluxo de potência. Adicionalmente, é apresentado em detalhes informações sobre: a implementação do modelo lógico orientado a objetos do CIM, numa base de dados relacional; os registros de equipamentos e a topologia de um trecho de linha de sub-transmissão aéreo e subterrâneo; as informações (exportadas e importadas no formato CIM/XML) inseridas na base de dados, e a geração do documento CIM/XML.

Palavras chaves: IEC 61970, CIM/XML, CIM RDF Schema

ABSTRACT

CUNHA JR., C. A. S. **Common Information Model/Extensible Markup Language (CIM/XML) to exchange data among Power Utilities' Control Centers**. 2005. 232 p. Dissertação Mestrado – Escola Politécnica, Universidade de São Paulo, São Paulo.

This dissertation provides an analysis about the application of a data model based on IEC 61970 (CIM) standard – a tool for exchanging operational and equipments information among different Electrical Power Utilities' computing systems. The purpose of this standard is to create a mechanism for information exchange using XML, called CIM/XML – a format specifically used by Electrical Power Utilities' Control Centers for data exchange. The IEC 61970 standard-based data models is presented as well the evaluation of CIM/XML as a tool for data interoperability among Electrical Power Utilities' databases, that uses different modeling and implementation approaches. The benefit of this model – besides of using a open standard technology (XML) that can be found on any type of computer – is the capability of store and transfer information not only from equipments, but also from network topology, load flow curves, generation scheduling, equipments outages, SCADA system measurements, status indication and alarms. It also enables the storage of simulation results, such as the power flow bus voltage and lines loads. Additionally, is also provided highly detailed information about: the CIM object-oriented model implementation mapped to a relational database; the records of equipments and topology of an aerial and underground subtransmission line section; the information (exported and imported using CIM/XML format) included in the database, and the CIM/XML document generation.

Key-words: IEC 61970, CIM/XML, CIM RDF Schema

SUMÁRIO

1. PREÂMBULO	14
1.1. MOTIVAÇÃO	14
1.1.1. <i>Cenário do problema</i>	14
1.1.2. <i>Solução</i>	15
1.1.2.1. Common Information Model (CIM)	15
1.1.2.2. eXtensible Markup Language (XML).....	16
1.1.3. <i>CIM – padrão internacional descrito na IEC 61970</i>	17
1.1.4. <i>CIM/XML – utilização obrigatória</i>	18
1.1.5. <i>CIM/XML – testes de interoperabilidade</i>	18
1.2. PROPOSTA.....	20
1.3. CONTRIBUIÇÃO.....	21
1.4. APRESENTAÇÃO.....	22
2. COMMON INFORMATION MODEL (CIM) [6]	24
2.1. INTRODUÇÃO	24
PACOTES DO CIM.....	25
2.2.1. <i>IEC 61970-301</i>	25
2.2.1.1. Pacote <i>Core</i>	25
2.2.1.2. Pacote <i>Domain</i>	26
2.2.1.3. Pacote <i>Generation</i>	26
2.2.1.4. Pacote <i>LoadModel</i>	26
2.2.1.5. Pacote <i>Meas</i>	26
2.2.1.6. Pacote <i>Outage</i>	27
2.2.1.7. Pacote <i>Protection</i>	27
2.2.1.8. Pacote <i>Topology</i>	27
2.2.1.9. Pacote <i>Wires</i>	27
2.2.2. <i>IEC 61970-302</i>	28
2.2.2.1. Pacote <i>Financial</i>	28
2.2.2.2. Pacote <i>Energy Scheduling</i>	28
2.2.2.3. Pacote <i>Reservation</i>	28
2.2.3. <i>IEC 61970-303</i>	29
2.2.3.1. Pacote <i>SCADA</i>	29
2.3. PRINCIPAIS CLASSES.....	30
2.4. APLICABILIDADE	32
2.5. VERSÕES	35
3. IMPLEMENTAÇÃO DO CIM NUMA BASE RELACIONAL	36
3.1. LEVANTAMENTO PRELIMINAR.....	36
3.2. INFORMAÇÕES E RECURSOS DISPONÍVEIS.....	37
3.3. A ESCOLHA DA LINGUAGEM DE PROGRAMAÇÃO	38
3.4. DIRETRIZES DE TRADUÇÃO DO MODELO LÓGICO PARA O MODELO RELACIONAL	38
3.5. CRIAÇÃO DA BASE RELACIONAL DO CIM (DETALHAMENTO DO PROCESSO)	42
3.6. CARGA DE DADOS	46
3.7. CONSULTAS REALIZADAS.....	53
4. DOCUMENTO CIM/XML.....	57
4.1. EXPORTAÇÃO DOS DADOS DA BASE RELACIONAL	57
4.2. VALIDAÇÃO DO DOCUMENTO CIM/XML	61

4.3. IMPORTAÇÃO DOS DADOS PARA A BASE RELACIONAL	62
5. CONCLUSÕES E RECOMENDAÇÕES	64
ANEXO A – IEC 61970 (DETALHAMENTO)	66
A.1. IEC 61970-301	66
<i>A.1.1. Pacote Core: Diagrama de Classes</i>	<i>66</i>
A.1.1.1. Pacote Core: Classe – BasePower	67
A.1.1.2. Pacote Core: Classe – BaseVoltage	67
A.1.1.3. Pacote Core: Classe – Bay	67
A.1.1.4. Pacote Core: Classe – ConductingEquipment	68
A.1.1.5. Pacote Core: Classe – ControlHouseEquipment	68
A.1.1.6. Pacote Core: Classe – CurveSchedData	68
A.1.1.7. Pacote Core: Classe – CurveSchedFormula	69
A.1.1.8. Pacote Core: Classe – CurveSchedule	69
A.1.1.9. Pacote Core: Classe – Equipment	70
A.1.1.10. Pacote Core: Classe – EquipmentContainer	71
A.1.1.11. Pacote Core: Classe – Naming	71
A.1.1.12. Pacote Core: Classe – PowerSystemResource	71
A.1.1.13. Pacote Core: Classe – PSRType	72
A.1.1.14. Pacote Core: Classe – SubControlArea	72
A.1.1.15. Pacote Core: Classe – Substation	72
A.1.1.16. Pacote Core: Classe – Terminal	72
A.1.1.17. Pacote Core: Classe – Unit	73
A.1.1.18. Pacote Core: Classe – VoltageLevel	73
<i>A.1.2. Pacote Domain</i>	<i>73</i>
A.1.2.1. Pacote Domain: Definição dos atributos das Classes	74
<i>A.1.3. Pacote Generation: Diagrama de Classes</i>	<i>79</i>
<i>A.1.4. Pacote Generation / GenerationDynamics: Diagramas de Classes</i>	<i>80</i>
A.1.4.1. Pacote Generation / GenerationDynamics: Classe – BWRSteamSupply	80
A.1.4.2. Pacote Generation / GenerationDynamics: Classe – CombustionTurbine	82
A.1.4.3. Pacote Generation / GenerationDynamics: Classe – CTTempMWCurve	83
A.1.4.4. Pacote Generation / GenerationDynamics: Classe – DrumBoiler	83
A.1.4.5. Pacote Generation / GenerationDynamics: Classe – FossilSteamSupply	83
A.1.4.6. Pacote Generation / GenerationDynamics: Classe – HeatRecoveryBoiler	85
A.1.4.7. Pacote Generation / GenerationDynamics: Classe – HydroTurbine	86
A.1.4.8. Pacote Generation / GenerationDynamics: Classe – PrimeMover	87
A.1.4.9. Pacote Generation / GenerationDynamics: Classe – PWRSteamSupply	87
A.1.4.10. Pacote Generation / GenerationDynamics: Classe – SteamSupply	89
A.1.4.11. Pacote Generation / GenerationDynamics: Classe – SteamTurbine	89
A.1.4.12. Pacote Generation / GenerationDynamics: Classe – Subcritical	90
A.1.4.13. Pacote Generation / GenerationDynamics: Classe – Supercritical	90
<i>A.1.5. Pacote Generation / Production: Diagrama de Classes</i>	<i>91</i>
A.1.5.1. Pacote Generation / Production: Classe – AccountBalance	93
A.1.5.2. Pacote Generation / Production: Classe – AirCompressor	94
A.1.5.3. Pacote Generation / Production: Classe – CAESPlant	94
A.1.5.4. Pacote Generation / Production: Classe – CogenerationPlant	94
A.1.5.5. Pacote Generation / Production: Classe – CombinedCyclePlant	95
A.1.5.6. Pacote Generation / Production: Classe – EmissionAccount	95
A.1.5.7. Pacote Generation / Production: Classe – EmissionCurve	96
A.1.5.8. Pacote Generation / Production: Classe – FossilFuel	96
A.1.5.9. Pacote Generation / Production: Classe – FuelAllocationSchedule	98
A.1.5.10. Pacote Generation / Production: Classe – GeneratingUnit	98
A.1.5.11. Pacote Generation / Production: Classe – GenUnitOpCostCurve	102

A.1.5.12.	Pacote <i>Generation / Production</i> : Classe – GenUnitOpSchedule	103
A.1.5.13.	Pacote <i>Generation / Production</i> : Classe – GrossToNetMWCurve	103
A.1.5.14.	Pacote <i>Generation / Production</i> : Classe – HeatInputCurve	103
A.1.5.15.	Pacote <i>Generation / Production</i> : Classe – HeatRateCurve	104
A.1.5.16.	Pacote <i>Generation / Production</i> : Classe – HydroGeneratingEfficiencyCurve	104
A.1.5.17.	Pacote <i>Generation / Production</i> : Classe – HydroGeneratingUnit	105
A.1.5.18.	Pacote <i>Generation / Production</i> : Classe – HydroPowerPlant	105
A.1.5.19.	Pacote <i>Generation / Production</i> : Classe – HydroPump	106
A.1.5.20.	Pacote <i>Generation / Production</i> : Classe – HydroPumpOpSchedule	107
A.1.5.21.	Pacote <i>Generation / Production</i> : Classe – IncrementalHeatRateCurve	107
A.1.5.22.	Pacote <i>Generation / Production</i> : Classe – InflowForecast	107
A.1.5.23.	Pacote <i>Generation / Production</i> : Classe – LevelVsVolumeCurve	108
A.1.5.24.	Pacote <i>Generation / Production</i> : Classe – PenstockLossCurve	108
A.1.5.25.	Pacote <i>Generation / Production</i> : Classe – Reservoir	108
A.1.5.26.	Pacote <i>Generation / Production</i> : Classe – ShutdownCurve	109
A.1.5.27.	Pacote <i>Generation / Production</i> : Classe – StartIgnFuelCurve	110
A.1.5.28.	Pacote <i>Generation / Production</i> : Classe – StartMainFuelCurve	110
A.1.5.29.	Pacote <i>Generation / Production</i> : Classe – StartRampCurve	110
A.1.5.30.	Pacote <i>Generation / Production</i> : Classe – StartupModel	110
A.1.5.31.	Pacote <i>Generation / Production</i> : Classe – SteamSendoutSchedule	112
A.1.5.32.	Pacote <i>Generation / Production</i> : Classe – TailbayLossCurve	112
A.1.5.33.	Pacote <i>Generation / Production</i> : Classe – TargetLevelSchedule	112
A.1.5.34.	Pacote <i>Generation / Production</i> : Classe – ThermalGeneratingUnit	112
A.1.6.	<i>Pacote LoadModel: Diagrama de Classes</i>	114
A.1.6.1.	Pacote <i>LoadModel</i> : Classe – AreaLoadCurve	114
A.1.6.2.	Pacote <i>LoadModel</i> : Classe – AreaLossCurve	115
A.1.6.3.	Pacote <i>LoadModel</i> : Classe – CustomerMeter	115
A.1.6.4.	Pacote <i>LoadModel</i> : Classe – DayType	115
A.1.6.5.	Pacote <i>LoadModel</i> : Classe – EquivalentLoad	115
A.1.6.6.	Pacote <i>LoadModel</i> : Classe – InductionMotorLoad	116
A.1.6.7.	Pacote <i>LoadModel</i> : Classe – LoadArea	116
A.1.6.8.	Pacote <i>LoadModel</i> : Classe – LoadDemandModel	116
A.1.6.9.	Pacote <i>LoadModel</i> : Classe – NonConformLoadSchedule	117
A.1.6.10.	Pacote <i>LoadModel</i> : Classe – PowerCutZone	117
A.1.6.11.	Pacote <i>LoadModel</i> : Classe – Season	117
A.1.6.12.	Pacote <i>LoadModel</i> : Classe – StationSupply	118
A.1.7.	<i>Pacote Meas: Diagrama de Classes</i>	119
A.1.7.1.	Pacote <i>Meas</i> : Classe – Control	121
A.1.7.2.	Pacote <i>Meas</i> : Classe – ControlType	122
A.1.7.3.	Pacote <i>Meas</i> : Classe – Limit	122
A.1.7.4.	Pacote <i>Meas</i> : Classe – LimitSet	123
A.1.7.5.	Pacote <i>Meas</i> : Classe – Measurement	123
A.1.7.6.	Pacote <i>Meas</i> : Classe – MeasurementType	125
A.1.7.7.	Pacote <i>Meas</i> : Classe – MeasurementValue	125
A.1.7.8.	Pacote <i>Meas</i> : Classe – MeasurementValueQuality	126
A.1.7.9.	Pacote <i>Meas</i> : Classe – MeasurementValueSource	126
A.1.7.10.	Pacote <i>Meas</i> : Classe – Quality61850	126
A.1.7.11.	Pacote <i>Meas</i> : Classe – ValueAliasSet	128
A.1.7.12.	Pacote <i>Meas</i> : Classe – ValueToAlias	128
A.1.8.	<i>Pacote Outage: Diagrama de Classes</i>	129
A.1.8.1.	Pacote <i>Outage</i> : Classe – ClearanceTag	129
A.1.8.2.	Pacote <i>Outage</i> : Classe – OutageSchedule	130
A.1.8.3.	Pacote <i>Outage</i> : Classe – SwitchingOperation	130
A.1.9.	<i>Pacote Protection: Diagrama de Classes</i>	132

A.1.9.1.	Pacote <i>Protection</i> : Classe – CurrentRelay	132
A.1.9.2.	Pacote <i>Protection</i> : Classe – ProtectionEquipment	133
A.1.9.3.	Pacote <i>Protection</i> : Classe – RecloseSequence	134
A.1.9.4.	Pacote <i>Protection</i> : Classe – SynchrocheckRelay	134
A.1.10.	Pacote <i>Topology</i> : <i>Diagrama de Classes</i>	136
A.1.10.1.	Pacote <i>Topology</i> : Classe – ConnectivityNode	137
A.1.10.2.	Pacote <i>Topology</i> : Classe – TopologicalIsland	137
A.1.10.3.	Pacote <i>Topology</i> : Classe – TopologicalNode	137
A.1.11.	Pacote <i>Wires</i> : <i>Diagrama de Classes</i>	139
A.1.11.1.	Pacote <i>Wires</i> : Classe – ACLineSegment	144
A.1.11.2.	Pacote <i>Wires</i> : Classe – Breaker	144
A.1.11.3.	Pacote <i>Wires</i> : Classe – BusBarSection	145
A.1.11.4.	Pacote <i>Wires</i> : Classe – Compensator	145
A.1.11.5.	Pacote <i>Wires</i> : Classe – CompositeSwitch	147
A.1.11.6.	Pacote <i>Wires</i> : Classe – Conductor	148
A.1.11.7.	Pacote <i>Wires</i> : Classe – ConductorType	149
A.1.11.8.	Pacote <i>Wires</i> : Classe – Connector	149
A.1.11.9.	Pacote <i>Wires</i> : Classe – DCLineSegment	149
A.1.11.10.	Pacote <i>Wires</i> : Classe – Disconnecter	150
A.1.11.11.	Pacote <i>Wires</i> : Classe – EnergyConsumer	150
A.1.11.12.	Pacote <i>Wires</i> : Classe – EquivalentSource	151
A.1.11.13.	Pacote <i>Wires</i> : Classe – Fuse	152
A.1.11.14.	Pacote <i>Wires</i> : Classe – Ground	152
A.1.11.15.	Pacote <i>Wires</i> : Classe – GroundDisconnecter	153
A.1.11.16.	Pacote <i>Wires</i> : Classe – HeatExchanger	153
A.1.11.17.	Pacote <i>Wires</i> : Classe – Jumper	153
A.1.11.18.	Pacote <i>Wires</i> : Classe – Junction	153
A.1.11.19.	Pacote <i>Wires</i> : Classe – Line	154
A.1.11.20.	Pacote <i>Wires</i> : Classe – LoadBreakSwitch	154
A.1.11.21.	Pacote <i>Wires</i> : Classe – MVarCapabilityCurve	154
A.1.11.22.	Pacote <i>Wires</i> : Classe – PowerTransformer	155
A.1.11.23.	Pacote <i>Wires</i> : Classe – RectifierInverter	155
A.1.11.24.	Pacote <i>Wires</i> : Classe – RegulatingCondEq	156
A.1.11.25.	Pacote <i>Wires</i> : Classe – RegulationSchedule	156
A.1.11.26.	Pacote <i>Wires</i> : Classe – StaticVarCompensator	157
A.1.11.27.	Pacote <i>Wires</i> : Classe – Switch	157
A.1.11.28.	Pacote <i>Wires</i> : Classe – SynchronousMachine	158
A.1.11.29.	Pacote <i>Wires</i> : Classe – TapChanger	161
A.1.11.30.	Pacote <i>Wires</i> : Classe – TransformerWinding	162
A.1.11.31.	Pacote <i>Wires</i> : Classe – VoltageControlZone	163
A.1.11.32.	Pacote <i>Wires</i> : Classe – WindingTest	163
A.1.11.33.	Pacote <i>Wires</i> : Classe – WireArrangement	165
A.1.11.34.	Pacote <i>Wires</i> : Classe – WireType	165
A.2.	IEC 61970-302	167
A.2.1.	Pacote <i>EnergyScheduling</i> : <i>Diagrama de Classes</i>	167
A.2.2.	Pacote <i>Financial</i> : <i>Diagrama de Classes</i>	168
A.2.3.	Pacote <i>Reservation</i> : <i>Diagrama de Classes</i>	169
A.3.	IEC 61970-303	170
A.3.1.	Pacote <i>SCADA</i> : <i>Diagrama de Classes</i>	170
A.3.1.1.	Pacote <i>SCADA</i> : Classe – CommunicationLink	171
A.3.1.2.	Pacote <i>SCADA</i> : Classe – RemoteControl	171
A.3.1.3.	Pacote <i>SCADA</i> : Classe – RemotePoint	171
A.3.1.4.	Pacote <i>SCADA</i> : Classe – RemoteSource	171
A.3.1.5.	Pacote <i>SCADA</i> : Classe – RemoteUnit	172

ANEXO B – TECNOLOGIA DA INFORMAÇÃO ENVOLVIDA.....	173
B.1. UML – UNIFIED MODELING LANGUAGE [13], [22], [23]	173
B.1.1. Definição	173
B.1.2. Objetivos.....	173
B.1.3. A Guerra dos Métodos.....	173
B.1.4. Histórico	175
B.1.5. Aceitação da UML.....	176
B.1.6. Padronização OMG.....	177
B.1.7. Aplicabilidade.....	177
B.1.8. Arquitetura de um Sistema	179
B.1.9. Desenvolvimento de Sistemas: Visão Tradicional ou Orientada a Objetos.....	182
B.1.10. Fases de Desenvolvimento de Sistemas.....	183
B.1.11. Elementos	186
B.1.12. Blocos de Construção.....	186
B.1.13. Itens	186
B.1.14. Relacionamentos.....	190
B.1.15. Diagramas	193
B.1.16. Visão Geral dos Diagramas	201
B.2. BANCO DE DADOS	202
B.2.1. Sistemas Gerenciadores de Bancos de Dados.....	202
B.2.2. Modelos de Dados	203
B.3. XML – EXTENSIBLE MARKUP LANGUAGE.....	206
B.3.1. Panorama Predecessor	206
B.3.1.1. Internet	206
B.3.1.2. HTML – Hiper Text Markup Language.....	208
B.3.2. Definição	212
B.3.3. Histórico	213
B.3.4. Visão Comparativa: HTML ou XML.....	214
B.3.5. Estrutura.....	216
B.3.6. Utilização e Importância.....	219
B.3.7. Padrões XML existentes no mercado	219
B.4. RDF E RDF SCHEMA	221
B.4.1. Conceito.....	221
B.4.2. Importação e Exportação das Informações de um Sistema de Potência	222
ANEXO C – TIPOS DE REATORES NUCLEARES.....	223
C.1. REATOR DE EBULIÇÃO DE ÁGUA (BWR – <i>BOILING WATER REACTOR</i>).....	223
C.2. REATOR DE ÁGUA PRESSURIZADA (PWR – <i>PRESSURIZED WATER REACTOR</i>)	226
REFERÊNCIAS	229

LISTA DE FIGURAS

Figura 1. Pacotes do CIM - adaptada de [2]	25
Figura 2. Diagrama de Classes básico do CIM - adaptada de [16]	30
Figura 3. Implementação das relações de herança na base relacional.....	40
Figura 4. Implementação do relacionamento “n:m” na base relacional	41
Figura 5. Representação gráfica da Linha de Transmissão SUL-BAN (circuitos 3 e 4).....	46
Figura 6. Carga realizada na tabela CIM.ACLineSegment	47
Figura 7. Carga realizada na tabela CIM.BaseVoltage	48
Figura 8. Carga realizada na tabela CIM.Breaker	48
Figura 9. Carga realizada na tabela CIM.ConductingEquipment.....	49
Figura 10. Carga realizada na tabela CIM.Conductor	49
Figura 11. Trecho final da carga realizada na tabela CIM.ConnectivityNode	50
Figura 12. Carga realizada na tabela CIM.Equipment	50
Figura 13. Carga realizada na tabela CIM.Line	51
Figura 14. Trecho inicial da carga realizada na tabela CIM.Naming	51
Figura 15. Carga realizada na tabela CIM.Switch.....	52
Figura 16. Carga realizada na tabela CIM.TopologicalNode.....	52
Figura 17. Consulta das propriedades elétricas dos segmentos da linha SUL-BAN 3-4	53
Figura 18. Representação gráfica dos segmentos da linha SUL-BAN 3-4	53
Figura 19. Consulta dos Nós Topológicos energizados e respectivos Nós de Conectividade	54
Figura 20. Representação gráfica dos Nós Topológicos energizados.	54
Figura 21. Consulta da localização, estado e Nó Topológico dos disjuntores	55
Figura 22. Representação gráfica da localização, estado e Nó Topológico dos disjuntores ...	56
Figura 23. Função Report do CIM Validator para o documento CIM/XML (export.xml)	61
Figura 24. Função Validate do CIM Validator para o documento CIM/XML (export.xml) ..	62
Figura 25. Diagrama de Classes do Pacote <i>Core</i> - [6].....	66
Figura 26. Diagrama de Classes do Pacote <i>Generation (main)</i> - [6].....	79
Figura 27. Diagrama de Classes do Pacote <i>Generation / GenerationDynamics</i> - [6].....	80
Figura 28. Diagrama de Classes do Pacote <i>Generation / Production (main)</i> - [6].....	91
Figura 29. Diagrama de Classes do Pacote <i>Generation / Production (hydro)</i> - [6]	92
Figura 30. Diagrama de Classes do Pacote <i>Generation / Production (thermal)</i> - [6]	93
Figura 31. Diagrama de Classes do Pacote <i>LoadModel</i> - [6]	114
Figura 32. Diagrama de Classes do Pacote <i>Meas (main)</i> - [6]	119
Figura 33. Diagrama de Classes do Pacote <i>Meas (measurements)</i> - [6]	120
Figura 34. Diagrama de Classes do Pacote <i>Meas (quality)</i> - [6]	121
Figura 35. Diagrama de Classes do pacote <i>Outage</i> - [6].....	129
Figura 36. Diagrama de Classes do Pacote <i>Protection</i> - [6].....	132
Figura 37. Diagrama de Classes do Pacote <i>Topology</i> - [6]	136
Figura 38. Diagrama de Classes do Pacote <i>Wires / InheritanceHierarchy</i> - [6]	139
Figura 39. Diagrama de Classes do Pacote <i>Wires / LineModel</i> - [6].....	140
Figura 40. Diagrama de Classes do Pacote <i>Wires / EquipmentContainment</i> - [6]	141
Figura 41. Diagrama de Classes do Pacote <i>Wires / TransformerModel</i> - [6].....	142
Figura 42. Diagrama de Classes do Pacote <i>Wires / RegulatingEquipment</i> - [6]	143
Figura 43. Diagrama de Classes do Pacote <i>Wires / VoltageControl</i> - [6]	144
Figura 44. Diagrama de Classes do pacote <i>EnergyScheduling</i> - [6]	167
Figura 45. Diagrama de Classes do pacote <i>Financial</i> - [6]	168
Figura 46. Diagrama de Classes do pacote <i>Reservation</i> - [6].....	169
Figura 47. Diagrama de Classes do pacote <i>SCADA</i> - [6]	170

Figura 48. A modelagem da arquitetura de um sistema - adaptada de [13]	180
Figura 49. Desenvolvimento de sistemas: Visão tradicional e Visão contemporânea	183
Figura 50. Representações gráficas de uma Classe - [22]	187
Figura 51. Exemplo da Classe Cliente - adaptada [22]	187
Figura 52. Representação gráfica de uma Interface - [13]	187
Figura 53. Representação gráfica de uma Colaboração - [13]	188
Figura 54. Representação gráfica de um Caso de Uso - [13]	188
Figura 55. Representação gráfica de um Componente - adaptada [13].....	188
Figura 56. Representação gráfica de um Nó - [13].....	189
Figura 57. Representação gráfica de uma Interação - [13].....	189
Figura 58. Representação gráfica de um Estado - [13].....	189
Figura 59. Representação gráfica de um Pacote - [13].....	190
Figura 60. Representação gráfica de uma Nota - [22].....	190
Figura 61. Representação gráfica de um Relacionamento de Dependência - [13].....	191
Figura 62. Representação gráfica de um Relacionamento de Associação - [13]	191
Figura 63. Representação gráfica de um Relacionamento de Agregação - [22]	192
Figura 64. Representação gráfica de um Relacionamento de Composição - [22].....	192
Figura 65. Representação gráfica de um Relacionamento de Generalização - [22].....	193
Figura 66. Representação gráfica de um Relacionamento de Realização - [13]	193
Figura 67. Exemplo de um Diagrama de Classes - [13].....	195
Figura 68. Exemplo de um Diagrama de Objetos - [13]	196
Figura 69. Exemplo de um Diagrama de Componentes - adaptada [13].....	197
Figura 70. Exemplo de um Diagrama de Implantação - [13]	198
Figura 71. Exemplo de um Diagrama de Caso de Uso - [13].....	199
Figura 72. Exemplo de um Diagrama de Atividade - [13]	199
Figura 73. Exemplo de um Diagrama de Sequência - [13]	200
Figura 74. Exemplo de um Diagrama de Colaboração - [13].....	200
Figura 75. Exemplo de um Diagrama de Gráfico de Estado - [13].....	201
Figura 76. Relacionamento entre os Diagramas da UML	201
Figura 77. Exemplo da Sintaxe HTML	211
Figura 78. HTML ou XML – (exemplo da sintaxe HTML).....	215
Figura 79. HTML ou XML – (exemplo da sintaxe XML)	215
Figura 80. <i>Browser</i> validando um documento XML.....	217
Figura 81. Sistema Reator de Ebulição de Água (<i>BWR System</i>) - [20].....	224
Figura 82. Sistema Reator de Água Pressurizada (<i>PWR System</i>) - [21].....	226

LISTA DE TABELAS

Tabela 1. Participantes/Produtos – 4º. teste de interoperabilidade do CIM/XML - [8]	19
Tabela 2. Empresas que utilizam o CIM e aplicações suportadas pelo modelo – [18]	32
Tabela 3. Aplicações integradas ou planejadas para a integração com o CIM – [18]	33
Tabela 4. Fornecedores envolvidos na implementação de aplicações do CIM – [18]	34
Tabela 5. CIM RDF Schema – definição do domínio BreakerConfiguration	42
Tabela 6. Saída gerada pelo programa REXX para o domínio BreakerConfiguration	43
Tabela 7. CIM RDF Schema – definição da classe <u>Bay</u>	43
Tabela 8. CREATE TABLE gerado pelo programa REXX para a Classe <u>Bay</u>	44
Tabela 9. Descrição da tabela Bay e respectivas colunas	45
Tabela 10. CIM RDF Schema – relacionamento entre as classes <u>Bay</u> e <u>VoltageLevel</u>	45
Tabela 11. Criação da chave estrangeira “MemberOf_VoltageLevel” na tabela Bay	46
Tabela 12. Cabeçalho fixo do documento CIM/XML	57
Tabela 13. Rodapé fixo do documento CIM/XML	57
Tabela 14. Trecho inicial do documento CIM/XML (exportação dos dados da base)	59
Tabela 15. Trecho final do documento CIM/XML (exportação dos dados da base)	60
Tabela 16. Trecho de comandos SQL para importação dos dados do documento CIM/XML	63
Tabela 17. Atributos das classes do pacote <i>Domain</i> .-[6]	74
Tabela 18. Representação de trecho de uma LT em CIM/XML	222

1. PREÂMBULO

1.1. Motivação

A troca de dados entre duas ou mais companhias do setor elétrico, num formato que seja acessível e inteligível por todos é um problema universal. Além disso, traduzir ou acessar os dados num formato adequado pelas aplicações em sistemas de potência usando formatos de dados proprietários, é um desafio [1].

1.1.1. Cenário do problema

Os fornecedores de *Energy Management Systems* (EMS) do mercado utilizam bancos e dicionários de dados de uso exclusivo. Cada fornecedor implementa seu sistema de forma distinta, embora todos apresentem basicamente as mesmas informações. Algumas vezes conseguimos importar dados de outros sistemas, mas raramente conseguimos exportar dados de maneira ampla para serem usados em sistemas diferentes. Esta restrição força o cliente a comprar novas aplicações do seu fornecedor original ou desenvolver, com sua própria equipe, soluções caseiras. É muito difícil outros fabricantes oferecerem produtos que possam ser facilmente integrados com as aplicações já instaladas.

Os modelos proprietários, a falta de flexibilidade, interoperabilidade e custos elevados das aplicações, foram aspectos relevantes que motivaram o *Electric Power Research Institute* (EPRI) a empreender, num esforço conjunto com os principais fabricantes e fornecedores de *softwares* para EMS norte americanos, o desenvolvimento do *Common Information Model* (CIM) necessário para um sistema de gerenciamento de energia [2].

1.1.2. Solução

1.1.2.1. Common Information Model (CIM)

Por volta de 1994, O EPRI começou um projeto chamado de *Control Center Application Program Interface* (CCAPI). Trabalhando com os melhores especialistas do mundo, o objetivo era eliminar a inflexibilidade e o alto custo oferecido pelos fabricantes de EMS sempre que novas aplicações eram necessárias. Geralmente, cada fornecedor de EMS não interoperava com aplicações de outros fornecedores por causa da engenharia especializada dos modelos de banco de dados, *softwares* proprietários e metodologias variadas. Neste esforço, surgiu o *Common Information Model* (CIM).

O CIM é um modelo padrão orientado a objetos para as companhias de eletricidade, no intuito de facilitar o desenvolvimento e integração dos sistemas aplicativos utilizados no planejamento, gerenciamento, operação e comercialização de energia elétrica [3].

Quando concluído em 1998, o nome CCAPI era bastante limitado visto que as necessidades, soluções e aplicações requeridas deste padrão foram além da competência exigida pelos centros de controle, pois o conteúdo do modelo não está limitado apenas aos centros de controle, mas atinge em profundidade toda a organização de uma companhia de energia elétrica. Outros esforços de modelagem são parte do CCAPI *work group* (WG), e todos são amplamente baseados na versão 10 do CIM (CIM 10 Version¹).

¹ CIM 10 Version, é uma revisão do CIM realizada pelo EPRI CCAPI Task Force e assim chamada nos Estados Unidos para distinguir das versões anteriores.

No projeto do *Control Center Application Program Interface* do EPRI, uma equipe formada por representantes das companhias de eletricidade, fornecedores de EMS, desenvolvedores de *softwares* de EMS e consultores, modelaram e criaram a especificação do CIM definindo que dados estão disponíveis e como eles são organizados. No CIM versão 10, a realização chave deste projeto é prover aos programas aplicativos uma visão dos sistemas de potência independentemente da plataforma. A equipe criou um modelo de referência que identificou interfaces requeridas para permitir a compatibilidade entre aplicações EMS e aplicações em distribuição, subestação, atendimento a clientes e outras áreas. Padrões adicionais fornecem todos os procedimentos de interface para trabalhar junto com o CIM para atingir a flexibilidade, modularidade de um projeto aberto [2].

Também em 1998, o *North American Electric Reliability Council* (NERC) começou a patrocinar uma série de reuniões no intuito de buscar um Modelo Comum para os Sistemas de Potência (*Common Power System Model* - CPSM). O propósito dessas reuniões era encontrar o modelo operacional para troca de informações entre as empresas do setor elétrico norte americano. Logo de início, o CIM do EPRI foi identificado como uma boa opção e isenta da influência de qualquer fornecedor para a modelagem das operações. O problema é que o CIM é um modelo abstrato; nem é uma especificação de modelagem de banco de dados, tampouco um formato de troca de informações.

1.1.2.2. eXtensible Markup Language (XML)

Durante este período, a *eXtensible Markup Language* (XML) surgiu como a tecnologia dominante para a codificação de documentos estruturados em novas aplicações. Por ser um padrão aberto, de fácil utilização, de simples aprendizado, um arquivo texto no padrão

Unicode UTF-8², que independente da plataforma e do sistema utilizado e principalmente por sua melhor característica, que é o conteúdo das informações contidas num arquivo XML, serem facilmente interpretadas por um programa aplicativo, tornaram a XML o formato de escolha do momento, para a troca de informações via Internet e outras redes privadas. Como resultado, um formato de troca baseado na definição dos dados do CIM e XML foi proposto pelo NERC e adotado como padrão [4].

1.1.3. CIM – padrão internacional descrito na IEC 61970

Em 2002, o CIM foi aprovado como um padrão internacional e detalhado pelo conjunto de normas IEC 61970, a qual define uma *Application Program Interface* (API) para um *Energy Management System* (EMS). Esta parte do padrão dentro do IEC é chamado de IEC 61970-301, a qual define o elemento principal do CIM do conjunto de pacotes³. Esses pacotes proporcionam uma visão lógica dos aspectos físicos de um EMS. A IEC 61970-303 define a visão lógica de um *Supervisory Control and Data Acquisition* (SCADA) [5] e a IEC 61970-501 define o *CIM Resource Description Framework* (RDF) *Schema* [6], que possibilita informações contidas num EMS serem convertidas para exportação, como um documento XML, o qual é chamado de documento CIM/XML. Isso permite a troca de modelos num formato padrão que qualquer EMS pode intercambiar via Internet ou rede privada.

² Padrão *Unicode* UTF-8, é um padrão de conjunto de caracteres com codificação em 16 bits, que tenta abranger caracteres de todas as principais linguagens escritas do mundo. UTF-8 é uma codificação de transformação que converte codificações de caracteres Unicode de modo que possam ser usadas em sistemas baseados em 8 bits, é a codificação de caracteres *default* da XML [7].

³ Pacote, é um mecanismo de propósito geral para organização de elementos em grupos (classes, componentes, etc.), utilizado na Linguagem Unificada de Modelagem (UML).

1.1.4. CIM/XML – utilização obrigatória

Por determinação do *North American Electric Reliability Council* (NERC), desde setembro de 2001, os *Security Coordination Centers* (SCCs) foram obrigados a utilizar o CIM/XML na troca de modelos, o que acarretou urgência no desenvolvimento de produtos para suportar o padrão [8].

1.1.5. CIM/XML – testes de interoperabilidade

O CIM traduzido em XML provê um importante padrão na troca de dados (arquivos) entre os sistemas de potência. O EPRI planejou uma série de testes de interoperabilidade entre produtos de diferentes fornecedores que validaram o uso e aceitação deste padrão [9], [10], [11].

O quarto teste de interoperabilidade do CIM/XML [8], realizado de 15 a 17 de julho de 2002, na cidade de San Francisco (CA), teve quatro participantes, conforme mostrado na Tabela 1. Várias amostras de arquivos modelos, de sistemas de potência, foram utilizadas. A ABB forneceu um modelo contendo 40 barramentos, a Siemens forneceu outro modelo contendo 100 barramentos. Além disso, modelos reais de grande escala fornecidos pela *Duke Energy* com 1.700 subestações e pela *California Independent System Operator* (CAISO) com aproximadamente 2.500 subestações, que foram utilizados nos testes de interoperabilidade anteriores estavam disponíveis. Atualizações incrementais e modelos parciais foram trocados e checados para validar a própria fusão dos arquivos modelos.

Tabela 1. Participantes/Produtos – 4º. teste de interoperabilidade do CIM/XML - [8]

Vendedor	Nome do Produto	Plataforma	Sist. Operac.
ABB	SABLE – Open technology system for implementation of Business Management and Energy Information systems.	COMPAQ Alpha server DS10, 600 MHZ	UNIX 5.0F
GE Network Solutions (1)	XA21	Sun Blade 100, 1 GB RAM	Solaris 2.8
GE Network Solutions (2)	Enterprise Gateway	IBM-compatible PC, 512 MB RAM	Windows 2000
Langdale Consulting	CIMBuilder – Knowledge Representation tool for managing CIM models.	X86-based PC. (Test unit configured with 700Mhz PIII, 196MB.)	Windows 2000
PTI	ODMS – Data Repository and Data Management System	IBM-compatible Laptop PC	Windows 2000

Os testes de interoperabilidade foram classificados em cinco categorias, como segue:

1. Teste básico de importação/exportação dos arquivos modelos – verificou a capacidade individual de cada produto na importação/exportação correta dos arquivos modelos dos sistemas de potência, baseados nos padrões CIM/XML.
2. Teste de interoperabilidade e transferência de dados de configuração do ICCP⁴ – verificou a capacidade do produto de um fornecedor em importar corretamente um modelo de dados previamente exportado por outro fornecedor que utiliza os padrões CIM/XML.
3. Teste de atualizações incrementais do modelo – Validou o uso dos novos padrões no envio de diferentes arquivos modelos.

⁴ ICCP (*Inter-Control Center Communications Protocol*), também conhecido com TASE.2 (*Tele-control application Service Element*), permite a troca de dados de tempo real entre centros de controle de uma companhia de energia elétrica e outros centros de controle, outras companhias, centros de controle regionais, etc.

4. Teste de transferência de um conjunto parcial de modelos – Exigiu que cada participante importasse um modelo parcial combinando-o com um modelo preexistente do mesmo fornecedor.
5. Teste de solução – Verificou o conteúdo correto dos arquivos modelos dos sistemas de potência, como também a troca e a transformação destes arquivos, incluindo a geração e carga – através da execução de aplicativos de fluxo de potência.

Atualmente, todos os principais fornecedores de Sistemas de Gerenciamento de Energia (EMS) expressam apoio pelo formato, tornando o modelo aberto para troca de informações uma realidade.

1.2. Proposta

O CIM é documentado com um conjunto de diagramas de classes⁵ usando UML *Unified Modeling Language*⁶. O CIM pode ser implementado como um banco de dados relacional, como um banco de dados orientado a objetos, ou pode ser utilizado para definir interfaces de componentes [12].

Conforme [13], (BOOCH; RUMBAUGH; JACOBSON, 2000) comentam que

[...] Em muitos sistemas cuja modelagem você fará, haverá objetos persistentes. Isso significa que poderão ser armazenados em um banco de dados para serem recuperados posteriormente. Com muita frequência, você usará um banco de dados relacional, um banco de dados orientado a objetos ou um banco de dados híbrido relacional/orientado a objetos para armazenamentos de itens persistentes. A UML é adequada para a modelagem de esquemas lógicos de bancos de dados, além dos próprios bancos de dados físicos.

Os diagramas de classes da UML são um superconjunto dos diagramas de entidade-

⁵ Diagrama de Classes, gráfico que mostra a estrutura do sistema : classes , tipos - seus conteúdos e relações.

⁶ UML, é um modelo de linguagem para modelagem de dados orientado a objetos. Sua modelagem visual favorece a compreensão e a documentação dos componentes do sistema e seus relacionamentos.

relacionamento (E-R), uma ferramenta básica de modelagem para o projeto lógico de banco de dados. Enquanto os diagramas E-R clássicos têm seu foco apenas nos dados, os diagramas de classes vão um pouco além, permitindo ainda a modelagem de comportamentos. No banco de dados físico, essas operações lógicas costumam servir como procedimentos armazenados ou para iniciar operações.

A escolha entre um modelo de banco de dados com arquitetura relacional, orientado a objetos ou mais recentemente objeto-relacional (“híbrido” de relacional com orientado a objetos), é uma questão de preferência, existem vantagens e desvantagens seja qual for a opção. Artigos sobre o assunto são facilmente encontrados na Internet. Porém, a discussão do tema no momento foge ao escopo deste trabalho.

No modelo relacional podemos destacar os seguintes aspectos: forte fundamentação matemática, tecnologia consolidada, maior número de usuários, diversos fabricantes, desempenho e confiabilidade.

Este trabalho aborda: *Common Information Model* (CIM), *Unified Modeling Language* (UML), *eXtensible Markup Language* (XML) e *Resource Description Framework* (RDF), com o objetivo prático de implementar o CIM numa base de dados relacional, carregar com alguns dados de uma concessionária de energia elétrica e descrever o processo de geração dos documentos CIM/XML utilizados na importação e exportação das informações armazenadas.

1.3. Contribuição

Este trabalho contribui nos seguintes aspectos:

1. No entendimento e utilização prática do *Common Information Model* (CIM) e tecnologia relacionada. O uso e domínio deste padrão internacional na troca de dados entre companhias de eletricidade significa a possibilidade de integração, via documentos CIM/XML, de aplicações distintas em plataformas heterogêneas e de diferentes

fabricantes. Devido às características inatas do CIM e XML, obtemos compatibilidade, flexibilidade, interoperabilidade e portabilidade no intercâmbio de informações de diferentes empresas acarretando economia de tempo e dinheiro.

2. Na implementação do modelo numa base de dados relacional. A linguagem SQL, padrão de acesso às bases relacionais, permite a implantação do CIM em qualquer base relacional ou objeto-relacional existente no mercado.
3. No entendimento prático do processo de importação e exportação de documentos CIM/XML para banco de dados relacionais ou objeto-relacionais.
4. Como material de referência e auxílio nas implementações de interfaces de importação e exportação de documentos CIM/XML em produtos já existentes e carentes desta funcionalidade.

1.4. Apresentação

Como visto, neste Capítulo 1 apresentou-se uma visão macro e do *Common Information Model/eXtensible Markup Language* (CIM/XML), o objetivo deste trabalho e as contribuições realizadas.

O Capítulo 2 mostra o *Common Information Model* (CIM). A descrição de cada pacote, sua finalidade e as informações nele contida.

No Capítulo 3 é abordado todo processo da implementação do *Common Information Model* (CIM) num Sistema Gerenciador de Banco de Dados Relacional (SGBDR) e respectiva carga de dados. As ferramentas utilizadas, técnicas e metodologias adotadas, são também comentadas.

O Capítulo 4 trata do processo de importação e exportação dos documentos CIM/XML sobre a base CIM relacional implementada, das técnicas e ferramentas utilizadas.

O Capítulo 5 contém as conclusões deste trabalho, destacando o poder de integração com flexibilidade e simplicidade, que as aplicações e sistemas relacionados ao setor de Energia Elétrica adquirem na utilização do CIM/XML.

O Capítulo 6 aborda as possibilidades de desenvolvimentos futuros com base no trabalho realizado.

O Anexo A apresenta o detalhamento do padrão IEC 61970 (*Common Information Model*), mostrando os diagramas de classes, respectivas descrições e atributos.

O Anexo B apresenta todo o embasamento teórico da *Unified Modeling Language* (UML), Base de Dados, *eXtensible Markup Language* (XML) e *Resource Description Framework* (RDF e RDF Schema). Este conjunto é a Tecnologia da Informação (TI) envolvida e necessária para atingir os objetivos deste trabalho.

O Anexo C apresenta os tipos de reatores nucleares que o modelo CIM trata. Este anexo é apenas um auxílio no entendimento das classes BWRSteamSupply e PWRSteamSupply descritas nos itens “A.1.4.1” e “A.1.4.9”, respectivamente.

2. COMMON INFORMATION MODEL (CIM) [6]

2.1. Introdução

Conforme comentado no primeiro capítulo, item 1.1.2.1, o CIM é um modelo padrão orientado a objetos para as companhias de eletricidade, no intuito de facilitar o desenvolvimento e integração dos sistemas aplicativos utilizados no planejamento, gerenciamento, operação e comércio de energia elétrica [3].

A maioria dos EMS (*Energy Management Systems*) e DMS (*Distribution Management Systems*) ainda são sistemas proprietários, o que dificulta a manutenção e o desenvolvimento de novas funcionalidades por terceiros, atualizações ou mudanças significativas nos sistemas normalmente são feitas pelos fornecedores originais ou por alguma equipe da casa (interna), acarretando desta forma custos elevados de manutenção.

O *Electric Power Research Institute (EPRI) Working Group on Control Center Application Interfaces (CCAPI)* tem por objetivo publicar um conjunto de normas e procedimentos para interfaces de aplicações, para o desenvolvimento de ferramentas de apoio e para promover o uso de *software* aberto em EMS e DMS. O CIM é a base de sustentação para a arquitetura do CCAPI.

O CIM é dividido em vários sub-modelos ou pacotes (*packages*) por questão de conveniência, onde os pacotes são agrupados para serem manipulados num único documento padrão, Figura 1.

2.2. Pacotes do CIM

Figura 1. Pacotes do CIM - adaptada de [2]

2.2.1. IEC 61970-301

A IEC 61970-301, define as classes fundamentais da rede elétrica e reúne os seguintes pacotes:

2.2.1.1. Pacote *Core*

O pacote *Core* contém as classes mais importantes do modelo, PowerSystemResource e ConductingEquipment (elas contém o cadastro e a topologia da rede) que são compartilhadas por todas aplicações do CIM. Outras classes importantes do pacote, não necessariamente são compartilhadas em todas as aplicações.

2.2.1.2. Pacote *Domain*

O pacote *Domain* é um dicionário de dados das quantidades e das unidades que definem os *datatypes* (tipos de dados) dos atributos (propriedades) que podem ser usados por qualquer classe em qualquer outro pacote.

2.2.1.3. Pacote *Generation*

O pacote *Generation* é composto pelos pacotes *Production* e *Generation Dynamics*. O pacote *Production* é responsável pelas classes que descrevem vários tipos de geradores (térmicos e hidráulicos). Estas classes também fornecem informações de custo de produção, que são utilizadas para alocar a demanda de maneira econômica entre as unidades empenhadas na geração; e no cálculo das quantidades de reserva. O pacote *Generation Dynamics* modela informações pertinentes aos reguladores, turbinas e caldeiras.

2.2.1.4. Pacote *LoadModel*

O pacote *LoadModel* é responsável pela modelagem dos consumidores de energia e carga do sistema, na forma de curvas e os dados associados às curvas. Circunstâncias especiais que podem afetar a carga, tal como as estações climáticas e o tipo do dia (fim de semana, dia útil e feriado), também são tratados neste pacote. Estas informações são utilizadas para previsão e gerenciamento de carga.

2.2.1.5. Pacote *Meas*

O pacote *Meas* contém as entidades que descrevem os dados dinâmicos das medições trocadas entre as aplicações.

2.2.1.6. Pacote *Outage*

O pacote *Outage* é uma extensão dos pacotes *Core* e *Wires*, e modela as informações de configuração da rede atual e planejada.

2.2.1.7. Pacote *Protection*

O pacote *Protection* é uma extensão dos pacotes *Core* e *Wires*, e modela as informações dos equipamentos de proteção como relés. Estas entidades são utilizadas nos simuladores de treinamento e nas aplicações de localização de faltas na rede de distribuição.

2.2.1.8. Pacote *Topology*

O pacote *Topology* é uma extensão do pacote *Core* que associada à classe Terminal modela a conectividade, que é a definição física de como os equipamentos encontram-se ligados entre si. Além disso, modela a topologia, que é a definição lógica de como o equipamento é conectado através do fechamento das chaves. A definição de topologia é independente de outras características elétricas.

2.2.1.9. Pacote *Wires*

O pacote *Wires* é uma extensão dos pacotes *Core* e *Topology*, modela informações sobre características elétricas das redes de transmissão e distribuição. Este pacote é utilizado por aplicações da rede, tais como: Estimador de Estado, Fluxo de Potência e Fluxo Ótimo de Potência.

2.2.2. IEC 61970-302

A IEC 61970-302 reúne os seguintes pacotes: *Financial*, *Energy Scheduling* e *Reservation*. Eles definem o modelo de dados para suportar a programação de carga e a comercialização de energia entre as companhias de eletricidade, em função da livre concorrência existente no mercado norte americano. Esta pode ser a mais difícil padronização, pois diferentes países possuem diferentes sistemas de regulamentação do setor [5].

2.2.2.1. Pacote Financial

O pacote *Financial* é responsável pelo faturamento e pagamento (liquidação da fatura). As classes que compõem este pacote representam as entidades legais envolvidas num contrato (acordo) formal ou informal.

2.2.2.2. Pacote Energy Scheduling

O pacote *Energy Scheduling* fornece a capacidade de programar e contabilizar as transações de troca de energia elétrica entre as companhias. Contém as transações de *megawatts* que são gerados, consumidos, perdidos, repassados, vendidos e comprados. Essas classes são utilizadas pela contabilidade para o faturamento da energia gerada, transmitida e serviços ancilares.

2.2.2.3. Pacote Reservation

O pacote *Reservation* contém informações das programações de intercâmbio de energia, ou seja: capacidade de geração, transmissão e serviços ancilares.

2.2.3. IEC 61970-303

A IEC 61970-303 trata das informações inerentes aos sistemas SCADA (*Supervisory Control and Data Acquisition*).

2.2.3.1. Pacote *SCADA*

O pacote *SCADA* descreve as medições, TPs (transformadores de potência), TCs (transformadores de corrente), UTRs (Unidade de Terminal Remota), *scan blocks* (blocos de varredura) e circuitos de comunicação. Suporta equipamentos de operação e controle, de telemetria, de aquisição de dados e alarmes. Os subtipos da telemetria deliberadamente combinam com as definições da UCA (*Utility Communication Architecture*) e com a especificação IEC 61850 (*Communication Networks and Systems Substations Part 7-3*) [5], [14], [15].

2.3. Principais Classes

O CIM é definido e mantido usando um conjunto de Diagramas de Classes UML (*Unified Modeling Language*). Uma visão das principais classes do CIM é mostrada na Figura 2.

Figura 2. Diagrama de Classes básico do CIM - adaptada de [16]

As classes mostradas na Figura 2, formam a base, o alicerce do CIM, de modo que possam suportar qualquer aplicação que necessite modelar o fluxo de eletricidade dos produtores aos consumidores [2], [16].

Os Diagramas de Classes de cada pacote do CIM são apresentados e descritos no ANEXO A, exceção feita às descrições das classes *Financial*, *Energy Scheduling* e *Reservation* (IEC 61970-302), maiores detalhes a respeito podem ser obtidos no endereço <http://cimuser.org> [6].

2.4. Aplicabilidade

A primeira versão do CIM data de 1995, e na sua evolução novas classes vêm sendo inseridas ao modelo para suportar as diversas aplicações que o setor elétrico demanda.

O trabalho publicado por *Robin Podmore* e *Marck Robisnson*, em novembro de 2000, intitulado: **Introduction to CIM** [18], relaciona algumas empresas de energia que tem implementado o CIM e as respectivas aplicações suportadas pelo modelo, conforme apresentado na Tabela 2.

Tabela 2. Empresas que utilizam o CIM e aplicações suportadas pelo modelo – [18]

EMPRESAS	APLICAÇÕES SUPORTADAS
<i>Allegheny Power</i>	<i>Operational Database Maintenance System</i>
<i>BC Hydro</i>	<i>Operator Training Simulator</i>
<i>BC Hydro</i>	<i>Voltage Stability Analysis</i>
<i>California ISO</i>	<i>Energy Management System Replacement</i>
<i>China Light and Power Company, Hong Kong</i>	<i>Data Repository System</i>
<i>Commonwealth Edison</i>	<i>Operator Training Simulator</i>
<i>Consolidated Edison</i>	<i>Feeder Management System</i>
<i>Consumers Energy</i>	<i>Advanced Network Applications</i>
<i>Duke Power</i>	<i>Operational Database Maintenance System</i>
<i>Entergy</i>	<i>Operational Database Maintenance System</i>
<i>ERCOT-ISO</i>	<i>Network Applications and EMS Replacement</i>
<i>Kansas City Power and Light</i>	<i>EMS, DMS and ICCP Integration.</i>
<i>Illinois Power</i>	<i>SCADA and AGC system</i>
<i>MAIN</i>	<i>Operational Database Maintenance System</i>
<i>NEPCO Jordan</i>	<i>CCAPI Open Application Platform</i>
<i>Southern Company Energy Marketing</i>	<i>ICCP Data Repository</i>
<i>Southwest Power Pool</i>	<i>Operational Database Maintenance System</i>
<i>Toronto Transit Commission</i>	<i>Operator Training Simulator</i>
<i>WAPA Watertown Operations</i>	<i>Advanced Network Applications</i>
<i>WAPA Electric Power Training Center</i>	<i>Operator Training Simulator</i>
<i>WAPA</i>	<i>Operational Database Maintenance System</i>
<i>EGAT</i>	<i>Operator Training Simulator</i>
<i>WSCC</i>	<i>Operator Training Simulator</i>

A Tabela 3, exibe as aplicações que foram integradas ou estão planejadas para integração com o CIM.

Tabela 3. Aplicações integradas ou planejadas para a integração com o CIM – [18]

APLICAÇÕES
<i>Data recording and playback</i>
<i>SCADA</i>
<i>AGC and Economic Dispatch</i>
<i>Interchange Scheduling</i>
<i>Operator Powerflow</i>
<i>State Estimator</i>
<i>Contingency Analysis</i>
<i>Security Constrained Optimal Powerflow</i>
<i>Operator Training Simulator</i>
<i>Voltage Stability Analysis</i>
<i>Dynamic Security Analysis</i>
<i>Available Transmission Capacity</i>
<i>Topology Processor</i>
<i>Topology Estimator</i>
<i>Unit Commitment</i>
<i>Interchange Distribution Calculator</i>
<i>Transaction Management System</i>
<i>Substation Maintenance Management</i>
<i>Distribution Powerflow</i>

A Tabela 4, exibe os fornecedores envolvidos na implementação de aplicações que utilizam o CIM.

Tabela 4. Fornecedores envolvidos na implementação de aplicações do CIM – [18]

FORNECEDORES	APLICAÇÕES
<i>ABB Systems Control</i>	<i>EMS Model Import and Export, Transmission Capacity Evaluation</i>
<i>ABB Network Management</i>	<i>EMS Open System Gateway</i>
<i>Alstom</i>	<i>Feeder Management System</i>
<i>Configured Energy Systems</i>	<i>EMS and DMS integration</i>
<i>CIM Logic</i>	<i>Model Maintenance Tools</i>
<i>Compaq</i>	<i>Business Integration Bus</i>
<i>Decision Systems International</i>	<i>Operator Training Simulator, Optimal Power Flow, Unit Commitment</i>
<i>Easi</i>	<i>Unbalanced Distribution Powerflow</i>
<i>ICL</i>	<i>Assess Management, Utility Systems Integration</i>
<i>Incremental Systems</i>	<i>Network Applications, Operator Training Simulator, EMS Model Maintenance and Conversion, Graphical User Interface, Data Recording and Playback</i>
<i>Mitsubishi Electric Corporation</i>	<i>SCADA, Transmission Management Applications, Distribution Management Applications.</i>
<i>OATI</i>	<i>Voltage Stability Analysis</i>
<i>Oracle</i>	<i>Data warehouse</i>
<i>OSI</i>	<i>Historical data recording</i>
<i>PowerData</i>	<i>Real-time relational database; JAVA graphical user interface, Network Applications, Operator Training Simulator, EMS Model Maintenance and Conversion</i>
<i>PowerTech</i>	<i>Voltage Stability Analysis</i>
<i>Psycor</i>	<i>EMS Model Maintenance and Conversion</i>
<i>Siemens</i>	<i>Network Applications, Dynamic Security Analysis</i>
<i>SISCO</i>	<i>Utility Integration Bus</i>
<i>Telegyr</i>	<i>Distribution Automation, Network Applications, Operator Training Simulator</i>
<i>Transdyn Controls</i>	<i>Operator Training Simulator</i>

2.5. Versões

O CIM versão 10 rev. 4 (CIMv10r4 – nov/2003), é uma versão estável e aprovada do modelo. Excluindo as classes do pacote *Domain*, o CIMv10r4 contém 174 classes, que compõem os 13 pacotes mencionados anteriormente.

No momento esforços são investidos nas versões CIMv10r5 Draft (jul/2004) e CIMv10r6 Draft (fev/2005), e estão disponíveis no endereço <http://cimuser.org> [6].

3. IMPLEMENTAÇÃO DO CIM NUMA BASE RELACIONAL

3.1. Levantamento Preliminar

No levantamento preliminar realizado sobre o modelo lógico do CIMv10r4 [6], obtivemos a seguinte estatística:

- **Classes:** 174
- **Atributos:** 1.806
- **Domínios:** 118
- **Associações (relacionamentos):** 335

Os números revelaram um volume de trabalho significativo para ser realizado manualmente, e mesmo sendo possível sua realização de forma não automatizada utilizando uma equipe de pessoas diretamente envolvidas nessa tarefa, os erros seriam inevitáveis.

Outro aspecto relevante e factível seria descobrir durante a fase de testes, que o método utilizado para implementar o modelo físico a partir do modelo lógico, não foi o mais apropriado, e refazer o trabalho, além do tempo adicional a ser gasto, seria uma tarefa pouco agradável.

A decisão de automatizar o processo de criação da base relacional do CIM, era um fato real.

As possibilidades em utilizar algum produto existente no mercado ou desenvolver ferramental, para que a tarefa de criação da base sofresse a mínima interferência manual possível, utilizando apenas as informações e recursos disponíveis, foram consideradas.

3.2. Informações e Recursos Disponíveis

As informações e recursos disponíveis para implementação da base relacional do CIM, utilizadas neste trabalho, foram:

1. O **CIM RDF Schema** [17].
2. A especificação IEC 61970 (CIM) e os diagramas de classes [2], [6].
3. As diretrizes para a tradução do Modelo Lógico para o Modelo de Dados Relacional [18].

Face as dificuldades em adquirir algum produto no mercado, que utilizasse como entrada de dados o **CIM RDF Schema**, processasse essas informações e gerasse a base de dados relacional, optamos pela codificação desta ferramenta.

Afinal, as informações disponíveis eram suficientes para a realização desta tarefa.

O **CIM RDF Schema**, por ser um arquivo XML, transparece algo pouco inteligível. Porém, é facilmente manipulado por qualquer linguagem de programação.

A especificação IEC 61970 (CIM), foi utilizada para contestar ou comprovar qualquer ensaio produzido pela ferramenta, durante a fase de codificação.

E finalmente, o mais importante, as diretrizes de como realizar a tradução do modelo lógico para o modelo relacional, nortearam a lógica de programação.

3.3. A escolha da Linguagem de Programação

A fluência numa linguagem conveniente para manipulação de caracteres, que não necessitasse de compilação (interpretada), com recursos que facilitassem a depuração em casos de erros de lógica, e pela expectativa dos diversos ensaios para exploração do conteúdo do **CIM RDF Schema**, optamos por utilizar a linguagem de programação REXX – ***RE**structured **eX**tended **eX**ecutor Language*.

Porém, qualquer linguagem que suporte leitura e gravação de arquivos, e tenha comandos de assinalamento, decisão e lógica repetitiva, poderia ser uma escolha apropriada.

3.4. Diretrizes de Tradução do Modelo Lógico para o Modelo Relacional

A criação da base de dados relacional do CIM, seguiu a maioria das recomendações expostas por *Robin Podmore* e *Marck Robinson* [18].

Conforme [18], a tradução da especificação do CIM para o âmbito dos bancos de dados relacionais, deve seguir as seguintes regras:

1. Toda classe do CIM possui uma tabela correspondente, de mesmo nome, na base relacional.
2. Todo atributo de uma classe possui uma coluna correspondente, de mesmo nome, na tabela relacionada.
3. Em cada tabela deve ser acrescentada uma coluna designada como chave primária (*primary key*) do tipo inteiro (*integer*), cujo nome é composto da concatenação do prefixo

“PK_” com o nome da tabela sendo criada. Por exemplo, a coluna **PK_PowerSystemResource** é chave primária incluída na tabela PowerSystemResource.

Observação: a implementação realizada neste trabalho a chave primária foi definida como sendo do tipo alfa-numérica de 30 posições (“*char(30)*”).

4. Exceto pela chave primária, todas as colunas da tabela podem admitir “NULL”.
5. As relações de herança, no modelo orientado a objeto do CIM, são tipicamente utilizadas para descrever objetos relacionados numa linhagem hierárquica. A implementação dessa hierarquia na base relacional é feita acrescentando um atributo comum a todos os membros dessa família.

O atributo comum a ser acrescentado a todos os membros da hierarquia, é: a chave primária da tabela raiz.

A Figura 3, mostra a implementação das relações de herança na base relacional.

Figura 3. Implementação das relações de herança na base relacional

Conforme o modelo de dados, a classe Naming é a raiz na hierarquia Naming, PowerSystemResource, Equipment, ConductingEquipment, Conductor e ALineSegment. Portanto, a chave primária **PK_Naming**, foi o atributo comum adicionado a cada entidade dessa estrutura.

6. O relacionamento de “muitos” para “muitos” (n:m), entre duas entidades deve ser implementado na base relacional com a criação de uma tabela adicional para representação dessa relação.

Este tipo de relacionamento é observado no Diagrama de Classes do pacote *SCADA*, entre as classes RemoteUnit (RTU) e CommunicationLink. A Figura 4, exibe a implementação recomendada na base relacional.

Figura 4. Implementação do relacionamento "n:m" na base relacional

3.5. Criação da Base Relacional do CIM (detalhamento do processo)

O detalhamento do processo de criação da base relacional do CIM, é apresentado neste tópico, mostrando os trechos do **CIM RDF Schema**, e as respectivas instruções SQL geradas pelos programas REXX.

Todas as instruções SQL apresentadas, foram geradas para o banco de dados MS SQL Server.

A Tabela 5, exibe o trecho do **CIM RDF Schema**, que define o domínio “BreakerConfiguration”.

Tabela 5. CIM RDF Schema – definição do domínio BreakerConfiguration

```
<rdfs:Class rdfs:about="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration">
  <rdfs:label xml:lang="en">BreakerConfiguration</rdfs:label>
  <rdfs:comment>Switching arrangement for Bay. </rdfs:comment>
  <cims:profile>Nerc</cims:profile>
  <cims:belongsToCategory rdfs:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Domain"/>
</rdfs:Class>
<rdf:Description rdfs:about="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration.singleBreaker">
  <rdfs:label xml:lang="en">singleBreaker</rdfs:label>
  <rdfs:comment/>
  <cims:profile>Nerc</cims:profile>
  <rdf:type rdfs:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration"/>
</rdf:Description>
<rdf:Description rdfs:about="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration.breakerAndAHalf">
  <rdfs:label xml:lang="en">breakerAndAHalf</rdfs:label>
  <rdfs:comment/>
  <cims:profile>Nerc</cims:profile>
  <rdf:type rdfs:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration"/>
</rdf:Description>
<rdf:Description rdfs:about="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration.doubleBreaker">
  <rdfs:label xml:lang="en">doubleBreaker</rdfs:label>
  <rdfs:comment/>
  <cims:profile>Nerc</cims:profile>
  <rdf:type rdfs:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration"/>
</rdf:Description>
<rdf:Description rdfs:about="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration.noBreaker">
  <rdfs:label xml:lang="en">noBreaker</rdfs:label>
  <rdfs:comment/>
  <cims:profile>Nerc</cims:profile>
  <rdf:type rdfs:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration"/>
</rdf:Description>
```

A Tabela 6, exibe a saída gerada pelo programa REXX no processamento do domínio “BreakerConfiguration”. A não implementação da restrição, de conter apenas os valores descritos, foi a opção adotada.

Tabela 6. Saída gerada pelo programa REXX para o domínio BreakerConfiguration

```
EXEC sp_addtype BreakerConfiguration, 'Char(14)';
```

A Tabela 7, exibe a definição da classe Bay no **CIM RDF Schema**.

Tabela 7. CIM RDF Schema – definição da classe Bay

```
<rdf:Class rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay">
  <rdf:label xml:lang="en">Bay</rdf:label>
  <rdf:comment>A collection of power system resources (within a given substation) including conducting
equipment, protection relays, measurements, and telemetry. The typeName indicates bay classification code
according to application, e.g., (OH = Overheadline) (CA = Capacitor).</rdf:comment>
  <cims:profile>Nerc</cims:profile>
  <cims:belongsToCategory rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Core"/>
  <rdf:subClassOf rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#EquipmentContainer"/>
</rdf:Class>
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.bayEnergyMeasFlag">
  <rdf:label xml:lang="en">bayEnergyMeasFlag</rdf:label>
  <rdf:comment>Indicates the presence/absence of kWh/kvarh measurements.</rdf:comment>
  <cims:profile>Nerc</cims:profile>
  <rdf:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
  <cims:dataType rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Boolean"/>
</rdf:Property>
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.bayPowerMeasFlag">
  <rdf:label xml:lang="en">bayPowerMeasFlag</rdf:label>
  <rdf:comment>Indicates the presence/absence of MW/Mvar measurements.</rdf:comment>
  <cims:profile>Nerc</cims:profile>
  <rdf:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
  <cims:dataType rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Boolean"/>
</rdf:Property>
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.breakerConfiguration">
  <rdf:label xml:lang="en">breakerConfiguration</rdf:label>
  <rdf:comment>Breaker configuration.</rdf:comment>
  <cims:profile>Nerc</cims:profile>
  <rdf:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
  <rdf:range rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BreakerConfiguration"/>
</rdf:Property>
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.busBarConfiguration">
  <rdf:label xml:lang="en">busBarConfiguration</rdf:label>
  <rdf:comment>Bus bar configuration.</rdf:comment>
  <cims:profile>Nerc</cims:profile>
  <rdf:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
  <rdf:range rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#BusbarConfiguration"/>
</rdf:Property>
```

A Tabela 8, exibe o CREATE TABLE gerado pelo programa REXX no processamento da classe Bay.

Tabela 8. CREATE TABLE gerado pelo programa REXX para a Classe Bay

```
CREATE TABLE CIM.Bay
(
  PK_Bay                Char(30)                NOT NULL
, bayEnergyMeasFlag     Bit
, bayPowerMeasFlag      Bit
, breakerConfiguration  BreakerConfiguration
, busBarConfiguration   BusbarConfiguration
, PK_Naming              Char(30)
, Primary Key (PK_Bay)
);
```

Comentários:

1. tipo “boolean” é correspondente ao tipo “bit” no MS SQL Server.
2. Os tipos “BreakerConfiguration” e “BusbarConfiguration” são domínios descritos no **CIM RDF Schema**.
3. “PK_Bay”, chave primária inserida conforme recomendação [18].
4. “PK_Naming”, chave primária da superclasse Naming inserida conforme recomendação [18]. A Classe Bay é subclasse de EquipmentContainer, que é subclasse de PowerSystemResource, que é subclasse de Naming.

As instruções SQL com a descrição da tabela e respectivas colunas, também foram gerados no processamento da classe Bay, conforme apresentado na Tabela 9.

As descrições traduzidas, das classes e atributos, foram extraídas do ANEXO A desta dissertação.

Tabela 9. Descrição da tabela Bay e respectivas colunas

```
-- sp_addextendedproperty da Tabela - Bay
exec sp_addextendedproperty N'MS_Description', N'É uma classe derivada de
EquipmentContainer. É uma coleção de recursos do sistema de potência,
dentro de uma subestação, incluindo equipamentos condutores, relés de
proteção, medições e telemetria.', N'user', N'CIM', N'table', N'Bay'
GO
exec sp_addextendedproperty N'MS_Description', N'indica a
presença/ausência de medições de kWh/kvarh.', N'User', N'CIM', N'table',
N'Bay', N'column', N'bayEnergyMeasFlag'
GO
exec sp_addextendedproperty N'MS_Description', N'indica a
presença/ausência de medições de MW/Mvar.', N'User', N'CIM', N'table',
N'Bay', N'column', N'bayPowerMeasFlag'
GO
exec sp_addextendedproperty N'MS_Description', N'expressa a configuração
do disjuntor.', N'User', N'CIM', N'table', N'Bay', N'column',
N'breakerConfiguration'
GO
exec sp_addextendedproperty N'MS_Description', N'expressa a configuração
do barramento.', N'User', N'CIM', N'table', N'Bay', N'column',
N'busBarConfiguration'
GO
```

A definição do relacionamento entre as classes Bay e VoltageLevel é descrito no **CIM RDF Schema**, conforme apresentado na Tabela 10.

Tabela 10. CIM RDF Schema – relacionamento entre as classes Bay e VoltageLevel

```
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#VoltageLevel.Contains_Bays">
<rdfs:label xml:lang="en">Contains_Bays</rdfs:label>
<rdfs:comment>The association is used in the naming hierarchy.</rdfs:comment>
<cims:profile>Nerc</cims:profile>
<rdfs:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#VoltageLevel"/>
<rdfs:range rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
<cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:0..n"/>
<cims:inverseRoleName rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.MemberOf_VoltageLevel"/>
</rdf:Property>
<rdf:Property rdf:about="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay.MemberOf_VoltageLevel">
<rdfs:label xml:lang="en">MemberOf_VoltageLevel</rdfs:label>
<rdfs:comment>The association is used in the naming hierarchy.</rdfs:comment>
<cims:profile>Nerc</cims:profile>
<rdfs:domain rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#Bay"/>
<rdfs:range rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#VoltageLevel"/>
<cims:multiplicity rdf:resource="http://iec.ch/TC57/1999/rdf-schema-extensions-19990926#M:0..1"/>
<cims:isAggregate>true</cims:isAggregate>
<cims:inverseRoleName rdf:resource="http://iec.ch/TC57/2001/CIM-schema-cim10#VoltageLevel.Contains_Bays"/>
```

Os relacionamentos no **CIM RDF Schema**, são traduzidos em instruções SQL para adição da chave estrangeira (*foreign key*) na tabela apropriada, conforme apresentado na Tabela 11.

Tabela 11. Criação da chave estrangeira “MemberOf_VoltageLevel” na tabela Bay

```
alter table CIM.Bay add MemberOf_VoltageLevel char(30);

alter table CIM.Bay add constraint Bay_MemberOf_VoltageLevel foreign key
(MemberOf_VoltageLevel) references CIM.VoltageLevel on delete no action;

exec sp_addextendedproperty N'MS_Description', N'FK --> VoltageLevel. The
association is used in the naming hierarchy.', N'User', N'CIM', N'table',
N'Bay', N'column', N'MemberOf_VoltageLevel'
```

3.6. Carga de Dados

Neste ponto do trabalho tínhamos as 174 classes do modelo lógico implementadas como tabelas na base relacional.

Para validação e teste do modelo relacional do CIM, escolhemos carregar as informações referentes a malha de interligação das subestações de sub-transmissão Sul e Bandeirantes à subestação da Eletropaulo Transformadora de Distribuição Canaã, cuja representação gráfica é mostrada na Figura 5.

Figura 5. Representação gráfica da Linha de Transmissão SUL-BAN (circuitos 3 e 4)

A representação gráfica da Linha de Transmissão SUL-BAN (circuitos 3 e 4), envolveu a inserção de dados em 16 tabelas da base MS SQL Server, pertencentes a 3 pacotes distintos do CIM.

Os pacotes envolvidos e respectivas classes que sofreram carga, foram:

- Pacote *Core*: (1) BaseVoltage, (2) ConductingEquipment, (3) Equipment, (4) EquipmentContainer, (5) Naming, (6) PowerSystemResource, (7) PSRType e (8) Substation.
- Pacote *Wires*: (9) ACLineSegment, (10) Breaker, (11) Conductor, (12) Line , (13) Switch e (14) WireType.
- Pacote *Topology*: (15) ConnectivityNode e (16) TopologicalNode.

A Figura 6, exibe a carga realizada na tabela “CIM.ACLineSegment”.

PK_ACLineSegment	PK_Naming	MemberOf_Line
1301-12011	1301-12011	1001
1301-12012	1301-12012	1001
1301-12021	1301-12021	1001
1301-12031	1301-12031	1001
1302-12011	1302-12011	1001
1302-12012	1302-12012	1001
1302-12021	1302-12021	1001
1302-12031	1302-12031	1001
*		

Figura 6. Carga realizada na tabela CIM.ACLineSegment

A Figura 7, exibe a carga realizada na tabela “CIM.BaseVoltage”.

PK_BaseVoltage	nominalVoltage	BasePower
1	88	<NULL>
2	138	<NULL>
3	145	<NULL>

Figura 7. Carga realizada na tabela CIM.BaseVoltage

A Figura 8, exibe a carga realizada na tabela “CIM.Breaker”.

PK_Breaker	ampRating	inTransitTime	PK_Naming	OperatedBy_ProtectionEquipments
1501	<NULL>	<NULL>	1501	<NULL>
1502	<NULL>	<NULL>	1502	<NULL>
1503	<NULL>	<NULL>	1503	<NULL>
1504	<NULL>	<NULL>	1504	<NULL>
1505	3150	<NULL>	1505	<NULL>
1506	3150	<NULL>	1506	<NULL>
1507	2500	<NULL>	1507	<NULL>
1508	2500	<NULL>	1508	<NULL>

Figura 8. Carga realizada na tabela CIM.Breaker

A Figura 9, exibe a carga realizada na tabela “CIM.ConductingEquipment”.

PK_ConductingEquipment	phases	PK_Naming	BaseVoltage	ProtectionEquipments
1301-12011	ABC	1301-12011	1	<NULL>
1301-12012	ABC	1301-12012	1	<NULL>
1301-12021	ABC	1301-12021	1	<NULL>
1301-12031	ABC	1301-12031	1	<NULL>
1302-12011	ABC	1302-12011	1	<NULL>
1302-12012	ABC	1302-12012	1	<NULL>
1302-12021	ABC	1302-12021	1	<NULL>
1302-12031	ABC	1302-12031	1	<NULL>
1501	ABC	1501	2	<NULL>
1502	ABC	1502	2	<NULL>
1503	ABC	1503	2	<NULL>
1504	ABC	1504	2	<NULL>
1505	ABC	1505	2	<NULL>
1506	ABC	1506	2	<NULL>
1507	ABC	1507	3	<NULL>
1508	ABC	1508	3	<NULL>

Figura 9. Carga realizada na tabela CIM.ConductingEquipment

A Figura 10, exibe a carga realizada na tabela “CIM.Conductor”

PK_Conductor	b0ch	bch	g0ch	gch	length	r	r0	x	x0	PK_Naming	ConductorType
1301-12011	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1301-12011	<NULL>
1301-12012	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1301-12012	<NULL>
1301-12021	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1301-12021	<NULL>
1301-12031	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1301-12031	<NULL>
1302-12011	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1302-12011	<NULL>
1302-12012	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1302-12012	<NULL>
1302-12021	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1302-12021	<NULL>
1302-12031	0	0	0	0	<NULL>	0,1043	0,4917	0,4256	1,7105	1302-12031	<NULL>

Figura 10. Carga realizada na tabela CIM.Conductor

A Figura 11, exibe trecho final da carga realizada na tabela “CIM.ConnectivityNode”.

SQL Server Enterprise Manager - [Data in Table 'ConnectivityNode' in 'CIM' on '{LOCAL}']

PK_ConnectivityNode	PK_Naming	TopologicalNode	MemberOf_EquipmentContainer
1302-12011/1	1302-12011/1	1606	1302-12011
1302-12011/2	1302-12011/2	1607	1302-12011
1302-12012/1	1302-12012/1	1607	1302-12012
1302-12012/2	1302-12012/2	1608	1302-12012
1302-12021/1	1302-12021/1	1609	1302-12021
1302-12021/2	1302-12021/2	1610	1302-12021
1302-12031/1	1302-12031/1	1607	1302-12031
1302-12031/2	1302-12031/2	1609	1302-12031
1501/1	1501/1	1608	1102
1502/1	1502/1	1603	1102
1503/1	1503/1	1601	1103
1504/1	1504/1	1606	1103
1505/1	1505/1	1604	1104
1506/1	1506/1	1609	1104
1507/1	1507/1	1605	1101
1508/1	1508/1	1610	1101

Figura 11. Trecho final da carga realizada na tabela CIM.ConnectivityNode

A Figura 12, exibe a carga realizada na tabela “CIM.Equipment”.

SQL Server Enterprise Manager - [Data in Table 'Equipment' in 'CIM' on '{LOCAL}']

PK_Equipment	PK_Naming	MemberOf_EquipmentContainer
1301-12011	1301-12011	<NULL>
1301-12012	1301-12012	<NULL>
1301-12021	1301-12021	<NULL>
1301-12031	1301-12031	<NULL>
1302-12011	1302-12011	<NULL>
1302-12012	1302-12012	<NULL>
1302-12021	1302-12021	<NULL>
1302-12031	1302-12031	<NULL>
1501	1501	1102
1502	1502	1102
1503	1503	1103
1504	1504	1103
1505	1505	1104
1506	1506	1104
1507	1507	1101
1508	1508	1101

Figura 12. Carga realizada na tabela CIM.Equipment

A Figura 13, exibe a carga realizada na tabela “CIM.Line”.

PK_Line	PK_Naming	TransmissionRightOfWay
1001	1001	<NULL>

Figura 13. Carga realizada na tabela CIM.Line

A Figura 14, exibe o trecho inicial da carga realizada na tabela “CIM.Naming”.

PK_Naming	aliasName	description
1001	LTA SUL-BAN 3-4	LINHA DE TRANSMISSAO AEREA
1101	ETD CAA	SUBESTACAO DE DISTRIBUICAO
1102	ETT BAN	SUBESTACAO DE TRANSMISSAO BANDEIRANTES
1103	ETT SUL	SUBESTACAO DE TRANSMISSAO SUL
1104	ETR CAA	SUBESTACAO DE TRANSICAO
1201	SUL BAN 3-4	RAMAL AEREO PRINCIPAL DA LT SUL BAN 3-4
12011	TRE 1	TRECHO 1 DO RAMAL SUL-BAN 3-4
12012	TRE 2	TRECHO 2 DO RAMAL SUL-BAN 3-4
1202	RSE CAA	RAMAL SUBTERRANEO PARA ETD CANAA
12021	TRE 1	TRECHO 1 DO RSE CANAA
1203	RAE CAA	RAMAL AEREO PARA ETR CANAA
12031	TRE 1	TRECHO 1 DO RAE CANAA
1301	CIRC 3	CIRCUITO 3 DA LTA SUL-BAN 3-4
1301-12011	SUL-BAN 3 / TRE 1	CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4
1301-12011/1	CN01 (SUL-BAN 3 / TRE 1)	CONNECTIVITY NODE 1 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4

Figura 14. Trecho inicial da carga realizada na tabela CIM.Naming

A Figura 15, exibe a carga realizada na tabela “CIM.Switch”.

PK_Switch	normalOpen	switchOnCount	switchOnDate	PK_Naming	SwitchingOperations	CompositeSwitch
1501	0	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1502	1	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1503	0	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1504	1	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1505	0	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1506	0	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1507	0	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>
1508	1	<NULL>	<NULL>	<NULL>	<NULL>	<NULL>

Figura 15. Carga realizada na tabela CIM.Switch

A Figura 16, exibe a carga realizada na tabela “CIM.TopologicalNode”.

PK_TopologicalNode	energized	loadCarrying	netInjectionMVar	netInjectionMW	observabilityFlag	phaseAngle	voltage	PK_Naming	TopologicalIsland
1601	1	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1601	<NULL>
1602	1	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1602	<NULL>
1603	0	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1603	<NULL>
1604	1	<NULL>	<NULL>	<NULL>	0	<NULL>	88	1604	<NULL>
1605	1	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1605	<NULL>
1606	0	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1606	<NULL>
1607	1	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1607	<NULL>
1608	1	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1608	<NULL>
1609	1	<NULL>	<NULL>	<NULL>	0	<NULL>	88	1609	<NULL>
1610	0	<NULL>	<NULL>	<NULL>	1	<NULL>	88	1610	<NULL>

Figura 16. Carga realizada na tabela CIM.TopologicalNode

Os dados inseridos nas onze tabelas mostradas anteriormente, são suficientes para a realização de algumas consultas na base relacional do CIM.

3.7. Consultas Realizadas

A Figura 17, exibe as linhas cadastradas, seus respectivos segmentos e propriedades elétricas.

SQL Query Analyzer - [Query - CTCSC.CIM.LANNT\ctcsc - C:\Consultas SQL no CIM (SQL Server)\Line & ALineSegment.sql]

```
-- Quais são as Linhas cadastradas, seus respectivos Segmentos e propriedades elétricas?
select N1.aliasname 'Nome da Linha'
, cast(A.PK_ALineSegment as char(10)) 'Segmento'
, N2.description 'Descrição do Segmento'
, cast(C.r as decimal(5,4)) 'r', cast(C.r0 as decimal(5,4)) 'r0'
, cast(C.x as decimal(5,4)) 'x', cast(C.x0 as decimal(5,4)) 'x0'
, CE.phases Fases, B.nominalVoltage 'Tensão'
from CIM.Line L, CIM.ALineSegment A, CIM.Conductor C
, CIM.ConductingEquipment CE, CIM.BaseVoltage B
, CIM.Naming N1, CIM.Naming N2
where L.PK_Line = A.MemberOf_Line
and A.PK_ALineSegment = C.PK_Conductor
and C.PK_Conductor = CE.PK_ConductingEquipment
and CE.BaseVoltage = B.PK_BaseVoltage
and L.PK_Naming = N1.PK_Naming
and A.PK_Naming = N2.PK_Naming
```

	Nome da Linha	Segmento	Descrição do Segmento	r	r0	x	x0	Fases	Tensão
1	LTA SUL-BAN 3-4	1301-12011	CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4	.1043	.4917	.4256	1.7105	ABC	88.0
2	LTA SUL-BAN 3-4	1301-12012	CIRCUITO 3 NO SEGUNDO TRECHO DO RAMAL SUL-BAN 3-4	.1043	.4917	.4256	1.7105	ABC	88.0
3	LTA SUL-BAN 3-4	1301-12021	CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUBTERRANEO CANAÃ	.1043	.4917	.4256	1.7105	ABC	88.0
4	LTA SUL-BAN 3-4	1301-12031	CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAÃ	.1043	.4917	.4256	1.7105	ABC	88.0
5	LTA SUL-BAN 3-4	1302-12011	CIRCUITO 4 NO PRIMEIRO TRECHO NO RAMAL SUL-BAN 3-4	.1043	.4917	.4256	1.7105	ABC	88.0
6	LTA SUL-BAN 3-4	1302-12012	CIRCUITO 4 NO SEGUNDO TRECHO NO RAMAL SUL-BAN 3-4	.1043	.4917	.4256	1.7105	ABC	88.0
7	LTA SUL-BAN 3-4	1302-12021	CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL SUBTERRANEO CANAÃ	.1043	.4917	.4256	1.7105	ABC	88.0
8	LTA SUL-BAN 3-4	1302-12031	CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAÃ	.1043	.4917	.4256	1.7105	ABC	88.0

Query batch completed. CTCSC (8.0) LANNT\ctcsc (52) CIM 0:00:00 8 rows Ln 1, Col 1

Figura 17. Consulta das propriedades elétricas dos segmentos da linha SUL-BAN 3-4

A Figura 18, exibe a representação gráfica dos segmentos da linha SUL-BAN 3-4.

Figura 18. Representação gráfica dos segmentos da linha SUL-BAN 3-4

A Figura 19, exibe os Nós Topológicos energizados e respectivos Nós de Conectividade.

SQL Query Analyzer - [Query - CTCSC.CIM.LANNT\ctcsc - C:\Consultas SQL no CIM (SQL Server)\TopologicalNode & ConnectivityNode.sql]

```
-- Quais são os Nós Topológicos energizados e seus respectivos Nós de Conectividade?
select T.PK_TopologicalNode 'Nó Topológico', N1.aliasname 'Desc.',
       C.PK_ConnectivityNode 'Nó de Conectividade', N2.description 'Descrição'
from CIM.TopologicalNode T, CIM.ConnectivityNode C, CIM.Naming N1, CIM.Naming N2
where T.PK_TopologicalNode = C.TopologicalNode
      and T.PK_Naming = N1.PK_Naming
      and C.PK_Naming = N2.PK_Naming
      and T.energized = 1
order by 1, 3 desc
```

	Nó Topológico	Desc.	Nó de Conectividade	Descrição
1	1601	TN01	1503/1	CONNECTIVITY NODE (DISJUNTOR NA ETT SUL PARA O CIRCUITO 3)
2	1601	TN01	1301-12011/1	CONNECTIVITY NODE 1 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4
3	1602	TN02	1301-12031/1	CONNECTIVITY NODE 1 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAA
4	1602	TN02	1301-12012/1	CONNECTIVITY NODE 1 DO CIRCUITO 3 NO SEGUNDO TRECHO DO RAMAL SUL-BAN 3-4
5	1602	TN02	1301-12011/2	CONNECTIVITY NODE 2 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4
6	1604	TN04	1505/1	CONNECTIVITY NODE (DISJUNTOR NA ETR CANAA PARA O CIRCUITO 3)
7	1604	TN04	1301-12031/2	CONNECTIVITY NODE 2 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAA
8	1604	TN04	1301-12021/1	CONNECTIVITY NODE 1 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUBTERRANEO CANAA
9	1605	TN05	1507/1	CONNECTIVITY NODE (DISJUNTOR NA ETD CANAA PARA O CIRCUITO 3)
10	1605	TN05	1301-12021/2	CONNECTIVITY NODE 2 DO CIRCUITO 3 NO PRIMEIRO TRECHO DO RAMAL SUBTERRANEO CANAA
11	1607	TN07	1302-12031/1	CONNECTIVITY NODE 1 DO CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAA
12	1607	TN07	1302-12012/1	CONNECTIVITY NODE 1 DO CIRCUITO 4 NO SEGUNDO TRECHO DO RAMAL SUL-BAN 3-4
13	1607	TN07	1302-12011/2	CONNECTIVITY NODE 2 DO CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL SUL-BAN 3-4
14	1608	TN08	1501/1	CONNECTIVITY NODE (DISJUNTOR NA ETT BANDEIRANTES PARA O CIRCUITO 4)
15	1608	TN08	1302-12012/2	CONNECTIVITY NODE 2 DO CIRCUITO 4 NO SEGUNDO TRECHO DO RAMAL SUL-BAN 3-4
16	1609	TN09	1506/1	CONNECTIVITY NODE (DISJUNTOR NA ETR CANAA PARA O CIRCUITO 4)
17	1609	TN09	1302-12031/2	CONNECTIVITY NODE 2 DO CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL AEREO CANAA
18	1609	TN09	1302-12021/1	CONNECTIVITY NODE 1 DO CIRCUITO 4 NO PRIMEIRO TRECHO DO RAMAL SUBTERRANEO CANAA

Query batch completed. CTCSC (8.0) LANNT\ctcsc (52) CIM 0:00:00 18 rows Ln 1, Col 1

Figura 19. Consulta dos Nós Topológicos energizados e respectivos Nós de Conectividade

A Figura 20, exibe a representação gráfica dos Nós Topológicos energizados.

Figura 20. Representação gráfica dos Nós Topológicos energizados.

Em função dos Nós Topológicos energizados, podemos afirmar que a ETT Sul está alimentando a ETD Canaã pelo circuito 3.

Outra maneira de verificar qual subestação de sub-transmissão está alimentando a ETD Canaã, seria verificando o estado dos disjuntores.

A Figura 21, exibe o estado dos disjuntores, sua localização e de qual Nó Topológico ele pertence.

SQL Query Analyzer - [Query - CTCSC.CIM.LANNT\ctcsc - C:\Consultas SQL no CIM (SQL Server)\Breaker & TopologicalNode.sql]

```
-- Qual o estado dos Disjuntores, onde estão localizados e de qual Nó Topológico ele pertence?
Select B.PK_Breaker 'Disjuntor', N1.aliasname 'Nome Usual', N1.description Descrição, 'Aberto' Estado
, N2.aliasname 'Nó Topológico'
from CIM.Breaker B, CIM.Switch S, CIM.Equipment E, CIM.ConnectivityNode C
, CIM.Naming N1, CIM.Naming N2
where B.PK_Breaker = S.PK_Switch and S.normalOpen = 1
and B.PK_Breaker = E.PK_Equipment
and E.PK_Equipment = substring(C.PK_ConnectivityNode,1,4)
and B.PK_Naming = N1.PK_Naming
and C.TopologicalNode = N2.PK_Naming
union all
Select B.PK_Breaker, N1.aliasname, N1.description Disjuntor, 'Fechado'
, N2.aliasname
from CIM.Breaker B, CIM.Switch S, CIM.Equipment E, CIM.ConnectivityNode C
, CIM.Naming N1, CIM.Naming N2
where B.PK_Breaker = S.PK_Switch and S.normalOpen = 0
and B.PK_Breaker = E.PK_Equipment
and E.PK_Equipment = substring(C.PK_ConnectivityNode,1,4)
and B.PK_Naming = N1.PK_Naming
and C.TopologicalNode = N2.PK_Naming
Order by 1
```

	Disjuntor	Nome Usual	Descrição	Estado	Nó Topológico
1	1501	TBANDJ04	DISJUNTOR NA ETT BANDEIRANTES PARA O CIRCUITO 4	Fechado	TN08
2	1502	TBANDJ03	DISJUNTOR NA ETT BANDEIRANTES PARA O CIRCUITO 3	Aberto	TN03
3	1503	TSULDJ03	DISJUNTOR NA ETT SUL PARA O CIRCUITO 3	Fechado	TN01
4	1504	TSULDJ04	DISJUNTOR NA ETT SUL PARA O CIRCUITO 4	Aberto	TN06
5	1505	RCAADJ03	DISJUNTOR NA ETR CANAÃ PARA O CIRCUITO 3	Fechado	TN04
6	1506	RCAADJ04	DISJUNTOR NA ETR CANAÃ PARA O CIRCUITO 4	Fechado	TN09
7	1507	DCAADJ03	DISJUNTOR NA ETD CANAÃ PARA O CIRCUITO 3	Fechado	TN05
8	1508	DCAADJ04	DISJUNTOR NA ETD CANAÃ PARA O CIRCUITO 4	Aberto	TN10

Successfully loaded query file C:\Consultas SQL no CIM (SQL Server)\Breaker & TopologicalNode.sql. CTCSC (8.0) LANNT\ctcsc (52) CIM 0:00:00 8 rows Ln 21, Col 11

Connections: 1

Figura 21. Consulta da localização, estado e Nó Topológico dos disjuntores

A consulta da Figura 21, confirma o fato da ETD Canaã estar sendo alimentada pelo circuito 3 da ETT Sul.

A representação gráfica da localização e estado dos disjuntores, e a qual Nó Topológico pertence, é exibida na Figura 22.

Figura 22. Representação gráfica da localização, estado e Nó Topológico dos disjuntores

4. DOCUMENTO CIM/XML

4.1. Exportação dos dados da Base Relacional

Em função dos nomes das tabelas e colunas da base relacional do CIM, corresponder exatamente aos nomes das classes e atributos especificados no modelo lógico, tornou o processo de exportação dos dados no formato CIM/XML, extremamente simples.

O documento CIM/XML é composto de três partes, sendo duas fixas que são o cabeçalho e o rodapé, e a parte variável que são os dados, as linhas inseridas em cada tabela.

A Tabela 12, exibe o cabeçalho fixo do documento CIM/XML.

Tabela 12. Cabeçalho fixo do documento CIM/XML

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:cim="http://iec.ch/TC57/2001/CIM-schema-cim10#">
```

A Tabela 13, exibe o rodapé fixo do documento CIM/XML.

Tabela 13. Rodapé fixo do documento CIM/XML

```
</rdf:RDF>
```

A geração da parte variável obedece as seguintes regras:

1. Somente serão exportadas as tabelas que contém dados. Para cada tabela (classe) fazemos a seguinte consulta “**SELECT * FROM CIM.nome_da_tabela**”, a tabela resultante desta consulta são todos os dados, linhas e colunas, inseridos na tabela selecionada.

2. Os campos “NULL” são ignorados.
3. O valor da chave primária, será concatenado com um prefixo e um sufixo, da seguinte maneira: “<cim:nome_tabela rdf:ID="valor_chave_primária"/>”. Por exemplo:
 <cim:ConductingEquipment rdf:ID="1501">.
4. Para cada valor da coluna **PK_Naming**, será realizado um “join” com a tabela **Naming** para se obter o conteúdo de **name** da tabela **Naming**, que será concatenado com um prefixo e um sufixo, da seguinte maneira:
 “<cim:Naming.name>Naming_name</cim:Naming.name>”. Por exemplo:
 <cim:Naming.name>1501</cim:Naming.name>.
5. O valor de cada coluna, que não é uma chave estrangeira e nem **PK_Naming**, será concatenado com um prefixo e um sufixo da seguinte maneira:
 “<cim:nome_tabela.nome_coluna>valor_coluna</cim:nome_tabela.nome_coluna>”. Por exemplo:
 <cim:ConductingEquipment.phases>ABC</cim:ConductingEquipment.phases>.
6. valor de cada chave estrangeira, será concatenado com um prefixo e um sufixo da seguinte maneira: “<cim:nome_tabela.nome_coluna rdf:resource="#valor_chave_estrangeira"/>”.
 Por exemplo: <cim:ConductingEquipment.BaseVoltage rdf:resource="#2"/>.
7. Para cada linha da tabela, inserir a “tag” de fechamento, da seguinte maneira:
 “</cim:nome_tabela>”. Por exemplo: </cim:ConductingEquipment>.

A Tabela 14, apresenta o trecho inicial do documento CIM/XML gerado para exportação dos dados da base relacional do CIM.

Tabela 14. Trecho inicial do documento CIM/XML (exportação dos dados da base)

```

<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:cim="http://iec.ch/TC57/2001/CIM-schema-cim10#">
  <cim:ACLineSegment rdf:ID="1301-12011">
    <cim:Naming.name>1301-12011</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1301-12012">
    <cim:Naming.name>1301-12012</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1301-12021">
    <cim:Naming.name>1301-12021</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1301-12031">
    <cim:Naming.name>1301-12031</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1302-12011">
    <cim:Naming.name>1302-12011</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1302-12012">
    <cim:Naming.name>1302-12012</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1302-12021">
    <cim:Naming.name>1302-12021</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:ACLineSegment rdf:ID="1302-12031">
    <cim:Naming.name>1302-12031</cim:Naming.name>
    <cim:ACLineSegment.MemberOf_Line rdf:resource="#1001"/>
  </cim:ACLineSegment>
  <cim:BaseVoltage rdf:ID="1">
    <cim:BaseVoltage.nominalVoltage>88.0</cim:BaseVoltage.nominalVoltage>
  </cim:BaseVoltage>
  <cim:BaseVoltage rdf:ID="2">
    <cim:BaseVoltage.nominalVoltage>138.0</cim:BaseVoltage.nominalVoltage>
  </cim:BaseVoltage>
  <cim:BaseVoltage rdf:ID="3">
    <cim:BaseVoltage.nominalVoltage>145.0</cim:BaseVoltage.nominalVoltage>
  </cim:BaseVoltage>
  <cim:Breaker rdf:ID="1501">
    <cim:Naming.name>1501</cim:Naming.name>
  </cim:Breaker>
  <cim:Breaker rdf:ID="1502">
    <cim:Naming.name>1502</cim:Naming.name>
  </cim:Breaker>
  <cim:Breaker rdf:ID="1503">
    <cim:Naming.name>1503</cim:Naming.name>
  </cim:Breaker>
  <cim:Breaker rdf:ID="1504">
    <cim:Naming.name>1504</cim:Naming.name>
  </cim:Breaker>
  <cim:Breaker rdf:ID="1505">
    <cim:Breaker.ampRating>3150.0</cim:Breaker.ampRating>
    <cim:Naming.name>1505</cim:Naming.name>
  </cim:Breaker>
  <cim:Breaker rdf:ID="1506">
    <cim:Breaker.ampRating>3150.0</cim:Breaker.ampRating>

```

A Tabela 15, apresenta o trecho final do documento CIM/XML gerado para exportação dos dados da base relacional do CIM.

Tabela 15. Trecho final do documento CIM/XML (exportação dos dados da base)

```

</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1604">
  <cim:TopologicalNode.energized>1</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>0</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1604</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1605">
  <cim:TopologicalNode.energized>1</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>1</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1605</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1606">
  <cim:TopologicalNode.energized>0</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>1</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1606</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1607">
  <cim:TopologicalNode.energized>1</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>1</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1607</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1608">
  <cim:TopologicalNode.energized>1</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>1</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1608</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1609">
  <cim:TopologicalNode.energized>1</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>0</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1609</cim:Naming.name>
</cim:TopologicalNode>
<cim:TopologicalNode rdf:ID="1610">
  <cim:TopologicalNode.energized>0</cim:TopologicalNode.energized>
  <cim:TopologicalNode.observabilityFlag>1</cim:TopologicalNode.observabilityFlag>
  <cim:TopologicalNode.voltage>88.0</cim:TopologicalNode.voltage>
  <cim:Naming.name>1610</cim:Naming.name>
</cim:TopologicalNode>
<cim:WireType rdf:ID="COD140-1">
  <cim:WireType.phaseConductorCount>1</cim:WireType.phaseConductorCount>
  <cim:WireType.phaseConductorSpacing>0.0</cim:WireType.phaseConductorSpacing>
  <cim:WireType.ampRating>770.0</cim:WireType.ampRating>
  <cim:WireType.gMR>3.0700000000E-2</cim:WireType.gMR>
  <cim:WireType.radius>3.0700000000E-2</cim:WireType.radius>
  <cim:WireType.resistance>2.7400000000E-2</cim:WireType.resistance>
  <cim:Naming.name>COD140-1</cim:Naming.name>
</cim:WireType>
</rdf:RDF>

```

4.2. Validação do documento CIM/XML

O documento CIM/XML exportado, foi validado com sucesso pelo **CIM Validator** [19].

A Figura 23, apresenta o resultado da função **Report** do **CIM Validator** para o documento CIM/XML (export.xml).

Figura 23. Função Report do CIM Validator para o documento CIM/XML (export.xml)

A Figura 24, apresenta o resultado da função **Validate** do **CIM Validator** para o documento CIM/XML (export.xml).

Figura 24. Função Validate do CIM Validator para o documento CIM/XML (export.xml)

4.3. Importação dos dados para a Base Relacional

O processo de importação é também muito simples.

A codificação do programa REXX para processar o documento CIM/XML gerado anteriormente, para a criação dos comandos SQL de inserção na base relacional do CIM não requer muito esforço de programação.

A Tabela 16, apresenta uma parte dos comandos SQL gerados pelo programa REXX, para importação dos dados do documento CIM/XML.

Tabela 16. Trecho de comandos SQL para importação dos dados do documento CIM/XML

```
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1301-12011',1001,'1301-12011')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1301-12012',1001,'1301-12012')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1301-12021',1001,'1301-12021')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1301-12031',1001,'1301-12031')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1302-12011',1001,'1302-12011')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1302-12012',1001,'1302-12012')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1302-12021',1001,'1302-12021')
GO
insert into CIM.ACLineSegment (PK_ACLineSegment,MemberOf_Line,PK_Naming)
values('1302-12031',1001,'1302-12031')
GO
insert into CIM.BaseVoltage (PK_BaseVoltage,nominalVoltage)
values('1',88.0)
GO
insert into CIM.BaseVoltage (PK_BaseVoltage,nominalVoltage)
values('2',138.0)
GO
insert into CIM.BaseVoltage (PK_BaseVoltage,nominalVoltage)
values('3',145.0)
GO
insert into CIM.Breaker (PK_Breaker,PK_Naming)
values('1501','1501')
GO
insert into CIM.Breaker (PK_Breaker,PK_Naming)
values('1502','1502')
GO
insert into CIM.Breaker (PK_Breaker,PK_Naming)
values('1503','1503')
GO
insert into CIM.Breaker (PK_Breaker,PK_Naming)
values('1504','1504')
GO
insert into CIM.Breaker (PK_Breaker,ampRating,PK_Naming)
values('1505',3150.0,'1505')
GO
insert into CIM.Breaker (PK_Breaker,ampRating,PK_Naming)
values('1506',3150.0,'1506')
GO
insert into CIM.Breaker (PK_Breaker,ampRating,PK_Naming)
values('1507',2500.0,'1507')
GO
```

5. CONCLUSÕES E RECOMENDAÇÕES

O trabalho aqui desenvolvido além da análise do modelo CIM, teve sua vertente prática e atingiu plenamente seu propósito, promovendo uma desmitificação do CIM.

A implementação da base relacional do CIM é explicada em detalhes, assim como o processo de importação e exportação das informações no formato CIM/XML.

A carga realizada com dados reais, mapearam os trechos de linha de sub-transmissão aéreo e subterrâneo que interliga as subestações transmissoras Sul-Bandeirantes à subestação da Eletropaulo transformadora de distribuição de energia, Canaã.

O intuito da carga de dados não foi validar o modelo lógico do CIM, como adequado para aplicações de cálculos elétricos, ou qualquer outra aplicação mencionada no item 2.4 deste trabalho, mas comprovar como adequado para armazenamento de dados de redes de energia elétrica. A carga teve o propósito real de gerar massa de dados para entender o processo de importação e exportação dos documentos CIM/XML, utilizados na troca de informações entre centros de controle de energia elétrica, através de sistemas EMS (*Energy Management Systems*) e DMS (*Distribution Management Systems*). Porém, a carga revelou que o processo de captura e conversão dos dados de uma base não implementada conforme o modelo lógico do CIM, é extremamente árduo.

Novas aplicações a serem desenvolvidas recomenda-se o uso do modelo especificado na IEC 61970 (CIM), pois além de ser um padrão internacionalmente reconhecido, sua utilização irá facilitar a troca de dados entre companhias de eletricidade e a integração, via documentos CIM/XML, de aplicações distintas em plataformas heterogêneas e de diferentes fabricantes.

Seria bastante interessante que os antigos EMS e DMS fossem remodelados por novas aplicações que implementassem o CIM. Porém, uma análise de custo e benefício dessa modificação deve ser considerada com muito critério.

O trabalho além das contribuições citadas no item 1.3, tornou disponível uma base de dados, compatível com o CIMv10r4, para exploração, comprovação, sugestão e contestação da real potencialidade do modelo. Isto possibilita que profissionais com conhecimentos tanto no contexto específico da engenharia elétrica quanto no uso de padrões abertos, insira o Brasil nas discussões e definições das novas versões do CIM.

Outra clara contribuição deste trabalho, foi a metodologia aplicada para criação da base relacional, o que possibilita a implementação física de qualquer versão do modelo lógico do CIM.

Como continuidade deste trabalho deixamos a sugestão de aproveitar esta experiência prática, para estimular o meio acadêmico ou empresas produtoras de *softwares* no desenvolvimento de uma ferramenta que possa ser comercializada. Na realidade seria a re-codificação dos programas REXX, na linguagem C ou qualquer outra linguagem apropriada para este propósito.

ANEXO A – IEC 61970 (DETALHAMENTO)

A.1. IEC 61970-301

A.1.1. Pacote Core: Diagrama de Classes

A Figura 25, apresenta o Diagrama de Classes do Pacote *Core*.

Figura 25. Diagrama de Classes do Pacote *Core* - [6]

A.1.1.1. Pacote *Core*: Classe – **BasePower**

A classe BasePower define a potência base dos equipamentos condutores. O atributo da classe é:

- **basePower**: definição da potência base em MVA.

A.1.1.2. Pacote *Core*: Classe – **BaseVoltage**

É um conjunto de tensões que são utilizadas para verificar a seção ou parte de um barramento.

A classe BaseVoltage e outros atributos de tensão no CIM, são fornecidos como um valor existente na coleção. O atributo da classe é:

- **nominalVoltage**: tensão base do recurso do sistema de potência (PowerSystemResource) em kV.

A.1.1.3. Pacote *Core*: Classe – **Bay**

É uma classe derivada de EquipmentContainer. É uma coleção de recursos do sistema de potência, dentro de uma subestação, incluindo equipamentos condutores, relés de proteção, medições e telemetria. Os atributos da classe são:

- **bayEnergyMeasFlag**: indica a presença/ausência de medições de kWh/kvarh.
- **bayPowerMeasFlag**: indica a presença/ausência de medições de MW/Mvar.
- **breakerConfiguration**: expressa a configuração do disjuntor *{singleBreaker, breakerAndHalf, doubleBreaker, noBreaker}*.

- **busBarConfiguration**: expressa a configuração do barramento *{singleBus, doubleBus, mainWithTransfer, ringBus}*.

A.1.1.4. Pacote Core: Classe – ConductingEquipment

Subclasse de Equipment. São os equipamentos do sistema de potência. ConductingEquipment está contido dentro de um EquipmentContainer e pode ser uma subestação (Substation), um nível de tensão (VoltageLevel) ou um Bay dentro de uma subestação. O atributo da classe é:

- **phases**: descreve as fases de um equipamento condutor. Os possíveis valores são: ABCN, ABC, ABN, ACN, BCN, AB, AC, BC, AN, BN, CN, A, B, C, N.

A.1.1.5. Pacote Core: Classe – ControlHouseEquipment

É uma subclasse de PowerSystemResource, são os equipamentos da casa de controle de uma subestação, cobre informações do tipo, alarmes de incêndio, temperatura ambiental, alarmes de porta e sobressalentes (spares). O atributo da classe é:

- **controlHouseEquipType**: tipo de equipamento da casa de controle.

A.1.1.6. Pacote Core: Classe – CurveSchedData

É uma classe derivada de Naming. Indica os valores dos pontos para a definição da curva ou programação (schedule). Os atributos da classe são:

- **rampData**: indica o valor da taxa de mudança da variável do eixo Y em relação à variável do eixo X.
- **rampDataValue**: indica o valor da taxa de mudança da variável do eixo Y em relação à variável do eixo X (0 = mudança instantânea).

- **xAxisData**: o valor da variável do eixo X, dependendo da unidade do eixo X.
- **y1AxisData**: o valor da primeira variável do eixo Y, dependendo da unidade do eixo Y.
- **y2AxisData**: o valor da Segunda variável do eixo Y (se presente), dependendo da unidade do eixo Y.

A.1.1.7. Pacote *Core*: Classe – CurveSchedFormula

É uma classe derivada de Naming. Contém a fórmula matemática para a definição da curva ou do segmento de programação. Os atributos da classe são:

- **xLowerBound**: indica o valor mínimo da variável X na escala da função.
- **xUpperBound**: indica o valor máximo da variável X na escala da função.
- **yFunction**: indica a fórmula matemática para a variável eixo Y com constantes numéricas.

A.1.1.8. Pacote *Core*: Classe – CurveSchedule

É uma classe derivada de Naming. Representa o relacionamento entre uma variável independente (eixo X) e uma ou duas variáveis dependentes (eixo Y1 e eixo Y2). As curvas podem também servir como programações (schedules). Os atributos da classe são:

- **curveStyle**: indica o estilo ou forma da curva {*constantYValue*, *rampYValue*, *straightLineYValues*, *formula*}.
- **rampMethod**: indica o deltaY versus o deltaX da unidade de medida. Aplica-se a todas as rampas (ramps).

- **rampStartMethod**: indica o método aplicado a rampa (ramp): 1. “0” do ponto inicial; 2. 50% do ponto inicial; 3. 100% do ponto inicial. Para os métodos 2 e 3, a rampa começa à frente do ponto inicial do eixo X (X-axis).

NOTA: para armazenamento, todas as rampas serão normalizadas ao método “1” (0 do ponto inicial).

- **rampUnits**: indica o deltaY versus o deltaX da unidade de medida, mesmo para “dois” valores de Y.
- **xAxisType**: indica o tipo da variável independente no eixo x {*absoluteTime*, *normalizedDailyTime*, *anyRealValue*}
- **xAxisUnits**: indica a unidade de medida do eixo X {*absoluteTime*, *normalizedTime*, *MW*}.
- **y1AxisUnits**: indica a unidade de medida do eixo Y1 {*unitOfCurrencyPerHour*, *MW*, *MVar*, *integer*}.
- **y2AxisUnits**: indica a unidade de medida do eixo Y2 {*MW*, *MVar*, *integer*}.
- **yAxisType**: indica o tipo da variável dependente {*singleYValue*, *twoYValues*}.

A.1.1.9. Pacote *Core*: Classe – Equipment

É uma subclasse de PowerSystemResource, são os dispositivos físicos, eletrônico ou mecânico, de um sistema de potência.

A.1.1.10. Pacote *Core*: Classe – EquipmentContainer

É uma subclasse de PowerSystemResource, é uma construção de modelagem utilizada para fornecer uma classe raiz para todas as classes de equipamentos (Equipment).

A.1.1.11. Pacote *Core*: Classe – Naming

É a classe raiz que fornece nomes comuns de atributos para todas as classes que necessitam de atributos de denominação (identificação). Os atributos da classe são:

- **aliasName**: identifica o nome do objeto ou uma ocorrência da classe (instância).
- **description**: armazena uma descrição do objeto ou uma ocorrência da classe (instância).
- **name**: nome único entre os objetos possuídos pelo mesmo pai.
- **pathName**: é uma concatenação de todos os nomes de cada recipiente (container)

A.1.1.12. Pacote *Core*: Classe – PowerSystemResource

Subclasse de Naming. Pode ser um item de um equipamento, tal como uma chave seccionadora (*switch*), um EquipmentContainer contendo vários itens de equipamentos (Equipment) como uma subestação (Substation), uma entidade organizacional como uma companhia de energia elétrica (Company) ou uma sub-região de controle (SubControlArea). A classe PowerSystemResource, serve para embutir uma coleção de recursos de um sistema de potência dentro de outros recursos. Por exemplo, uma chave seccionadora poderia ser um membro de uma subestação (Substation) e a subestação poderia ser um membro de uma divisão de uma companhia (Company).

A.1.1.13. Pacote *Core*: Classe – PSRType

- É uma classe derivada de Naming. É o classificador das ocorrências (instâncias) da mesma classe, por exemplo: trechos de linha de corrente alternada aéreas ou subterrâneas. Este mecanismo de classificação pretende prover flexibilidade para o que está fora do escopo deste padrão, isto é, prover uma adaptação para o que não é padrão.

A.1.1.14. Pacote *Core*: Classe – SubControlArea

É uma classe derivada de PowerSystemResource, e define uma área com a finalidade de intercambiar informações com regiões circunvizinhas, que pode ou não, servir como uma área de controle.

A.1.1.15. Pacote *Core*: Classe – Substation

É uma subclasse de EquipmentContainer, e consiste de um conjunto de equipamentos para tratar a energia elétrica, nela transitada, com a finalidade de chaveamento (switching) ou de modificação de suas propriedades.

A.1.1.16. Pacote *Core*: Classe – Terminal

É uma classe derivada de Naming, e representa um ponto elétrico de conexão a uma parte do equipamento condutor. Os terminais são ligados em pontos de conexão física, chamados nós de conectividade (connectivity nodes).

A.1.1.17. Pacote *Core*: Classe – Unit

Subclasse de Naming. Representa a quantidade que está sendo medida. O **Unit.name** será único entre todas as quantidades especificadas, e descreverá a unidade. O **Unit.aliasName** é utilizado para a localização do objeto. Os atributos são os mesmos da superclasse.

A.1.1.18. Pacote *Core*: Classe – VoltageLevel

É uma classe derivada de EquipmentContainer. É uma coleção de equipamentos em uma tensão comum do sistema formando um *switchgear*. Os equipamentos são tipicamente disjuntores, barramentos, dispositivos de instrumentação, de controle, de proteção e regulação ou uma montagem deles todos. Os atributos da classe são:

- **highVoltageLimit**: o limite de máxima tensão do barramento em kV.
- **lowVoltageLimit**: o limite de mínima tensão do barramento em kV.

A.1.2. Pacote *Domain*

Este pacote contém a definição dos *datatypes* primitivos, inclusive das unidades de medida e dos valores admissíveis. Cada *datatype* contém um atributo de valor e uma unidade opcional de medida, a qual é especificada como uma variável estática inicializada conforme descrição textual da unidade de medida. Os valores permitidos para as enumerações estão relacionados na documentação do atributo usando a sintaxe de restrição da UML (entre chaves – {}). Os tamanhos das cadeias de caracteres são relacionados na descrição do texto onde requeridos.

A.1.2.1. Pacote *Domain*: Definição dos atributos das Classes

A Tabela 17, apresenta todos os atributos das classes do pacote *Domain* e respectivas definições.

Tabela 17. Atributos das classes do pacote *Domain*.-[6]

ATRIBUTOS		DEFINIÇÃO
01	AbsoluteDateTime	Data e Hora em “yyy-mm-ddThh:mm:ss”, conforme ISO 8601 usando a forma estendida e UTC (<i>Coordinated Universal Time</i>) no lugar das zonas de tempo locais.
02	ActivePower	Produto do valor RMS da tensão e do valor RMS do componente da corrente em fase (Megawatt - MW)
03	Admittance	Admitância, razão da corrente sobre a tensão, expressa Siemens
04	AngleDegrees	Medida do ângulo em graus
05	AngleRadians	Ângulo da fase em radianos
06	ApparentPower	Produto do valor RMS da tensão e do valor do RMS da corrente (MegaVoltAmpères – MVA)
07	BoilerControlMode	Caldeira {Following, Coordinated}
08	Boolean	Booleano
09	BreakerConfiguration	Configuração de chaveamento para um Bay {singleBreaker, breakerAndHalf, doubleBreaker, noBreaker}
10	BusbarConfiguration	Disposição do barramento para a Bay {singleBus, doubleBus, mainWithTransfer, ringBus}
11	Classification	1..n, sendo 1: o mais detalhado, ou a mais alta prioridade, etc.
12	ClearanceTagType	Tipo de trabalho a ser executado e/ou o tipo de controle de supervisão.
13	CompanyType	Tipo de companhia {pool, municipal, private}
14	CompensatorType	Tipo do compensador {shunt, series}
15	Complex	Complexo
16	CompositeSwitchType	Um código alfanumérico que pode ser usado como uma referência à informação extra, tal como a descrição do esquema inter-travamento, se houver algum
17	Conductance	O fator pelo qual a tensão deve ser multiplicada para dar a perda de potência correspondente do circuito. Parte real da admitância, expressa em Siemens
18	ControlHouseEquipmentType	Tipo de equipamento da casa do controle
19	ControlMode	Nome textual para uma modalidade de controle
20	CoolantType	Método de refrigeração de uma máquina {air, hydrogenGas, water}

continua

continuação

ATRIBUTOS		DEFINIÇÃO
21	CostPerEnergyUnit	Custo, em unidades da moeda corrente, por MWh de eletricidade gerada (\$/MWh)
22	CostPerHeatUnit	Custo, em unidades da moeda corrente, por milhões de Btus do calor gerado (\$/MBtu)
23	CostPerHour	Custo, em unidades da moeda corrente, por hora de tempo decorrido (\$/Hora)
24	Counter	Medida de quantidade (deve ser inteiro igual ou maior do que zero)
25	CurrentFlow	Fluxo de corrente - valor positivo indica: sai do <u>ConductingEquipment</u> entra no <u>ConnectivityNode</u> - (Ampères)
26	CurveStyle	Estilo ou forma da curva {constantYValue, rampYValue, straightLineYValues, formula}
27	Damping	Variação por unidade de megawatt por unidade de frequência (PU_MW/PU_Frequency)
28	DateTime	Data e hora
29	DayTypeName	Grupo de dias similares {weekday, weekend, holiday}
30	Double	Ponto flutuante de dupla precisão
31	Emission	Quantidade de emissão pelo índice de calor do combustível (kg/MBtu)
32	EmissionType	O tipo de emissão {hydrogenSulfide, carbonDioxide, carbonDisulfide, sulfurDioxide, nitrogenDioxide, chlorine}
33	EmissionValueSource	A origem do valor da emissão {measured, calculated}
34	EnergyAsMWh	Energia elétrica, em megawatt/hora (MWh)
35	EnumeratedType	Um tipo numerável para os atributos cujo domínio é um conjunto finito de valores. Os valores válidos são normalmente listados na descrição do atributo, usando a sintaxe de restrições da UML {valor1, valor2... valorN}.
36	ExcitingCurrent	A corrente de excitação no teste de circuito aberto, expresso como uma percentagem da corrente contínua, na tensão nominal
37	Exponent	Valor do expoente
38	Fraction	Valor fracionário. A soma de todas as frações da mesma quantidade deve ser 1.0
39	FreqBiasFactor	Fator de polarização da frequência da área de controle (MW/0.1 Hz)
40	Frequency	Ciclos por segundo (Hertz)
41	FuelType	Tipo de combustível {coal, oil, gas}
42	GeneratorControlMode	Modo de controle da unidade, Setpoint ou Pulse {S, P}
43	GeneratorControlSource	A fonte dos controles da unidade geradora {Unavailable, Off-AGC, On -AGC, Plant Control}
44	GeneratorOperatingMode	Modo de operação do gerador secundário {Off, Manual, Fixed, LFC, AGC, EDC, RPN, MRN, REG}
45	HeatPerHour	O calor gerado, em milhões de Btus, por hora do tempo decorrido (MBtu/Hora)

continua

continuação

ATRIBUTOS		DEFINIÇÃO
46	Hours	Tempo, em horas
47	HydroPlantType	O tipo de usina hidrelétrica {runOfRiver, pumpedStorage, majorStorage, minorStorage}
48	Impedance	Impedância, razão da tensão sobre a corrente, expressa em ohms
49	Inductance	Indutância, em milihenries
50	Inertia	Megawatt-segundos pela taxa de MegaVolt-Ampères da unidade referenciada no sistema base MVA. Valores típicos entre 3.0 - 5.0 (PU_MW-segundos)
51	Load	Carga em MW, maior do que 0
52	LoadLoss	A perda no teste de curto-circuito na corrente nominal (kW)
53	LongDouble	
54	LongLength	Unidade longa de extensão (por exemplo, milha, quilômetro)
55	LongLong	
56	Money	Quantia de dinheiro (em unidade da moeda corrente - \$)
57	NoLoadLoss	A perda no teste de circuito aberto na tensão nominal (kW)
58	Numeric	Númerico {INT_TYPE, UNSIGNED_TYPE, DOUBLE_TYPE, COMPLEX_TYPE, DATE_TIME_TYPE, ULONG_LONG_TYPE, SHORT_TYPE, LONG_LONG_TYPE, LONG_DOUBLE_TYPE, BOOLEAN_TYPE}
59	NumericType	Tipo numérico {INT_TYPE, UNSIGNED_TYPE, DOUBLE_TYPE, COMPLEX_TYPE, DATE_TIME_TYPE, ULONG_LONG_TYPE, SHORT_TYPE, LONG_LONG_TYPE, LONG_DOUBLE_TYPE, BOOLEAN_TYPE}
60	OperatingMode	Nome do modo operacional
61	ParticipationFactor	Fator de participação econômica da unidade geradora
62	PenaltyFactor	Definido como: $1 / (1 - \text{Perda de Transmissão Incremental})$; com a Perda de Transmissão Incremental expressa como um valor mais positivo ou negativo. A escala típica de fatores de perda é (0.9 a 1.1)
63	PenstockType	Tipo de comporta da hidrelétrica
64	PerCent	Normalmente 0 - 100 em uma base definida
65	PhaseCode	Coleção de identificadores de fase {ABCN, ABC, ..., A, B, C, N}
66	PowerFactor	Fator de Potência - relação entre o ActivePower e o ApparentPower
67	PowerROCPeMin	Taxa de mudança de megawatt por o minuto (MW/minuto)
68	PowerROCPeSec	Taxa de mudança de megawatt por o segundo (MW/segundo)
69	PowerVersusFrequency	Unidade de variação real de potência pela unidade de variação de frequência (PU_MW/PU_Frequency)
70	PowerVersusVoltage	Unidade de variação real de potência pela unidade de variação de tensão (PU_MW/PU_kV)
71	Pressure	Pressão expressa em libra-força por polegada quadrada (psi ou lbf/in2)

continua

continuação

ATRIBUTOS		DEFINIÇÃO
72	Priority	Define níveis de prioridade com um valor numérico, sendo zero o nível de prioridade mais elevada
73	PU	Por Unidade - um valor positivo ou negativo referente a uma base definida. Os valores típicos variam de -10 a +10
74	PUkVPerMVar	Variação da tensão pela potência reativa (PU_kV/MVAr)
75	RampMethod	O deltaY versus deltaX das unidades de medida {yUnitsPerMinute, rampCompletionInMinutes}
76	RampStartMethod	O método de aplicação à rampa {oneHundredPerCent, fiftyPerCent, zero}
77	RampUnits	O deltaY versus deltaX das unidades de medida {MWPerMinute, rampCompletionInMin}
78	RateOfChange	Taxa de mudança de variável sem dimensão
79	Ratio	Relação de dois valores com as mesmas unidades
80	Reactance	Reatância (parte imaginária da impedância), em ohms, na frequência calculada
81	ReactivePower	Produto do valor RMS da tensão e do valor RMS do componente da quadratura da corrente (MegaVoltAmpères Reativo - MVAr)
82	Reference	Referência ao texto externo
83	RemoteUnitType	Tipo da Unidade Remota {RTU, SubstationControlSystem, ControlCenter}
84	Resistance	Resistência (parte real da impedância), em ohms
85	SeasonName	Estação climática {winter, spring, summer, fall}
86	Seconds	Tempo, em segundos
87	Short	
88	ShortLength	Unidade curta de comprimento (por exemplo pés; metro)
89	Source	A fonte dá a informação relacionada à origem de um valor. O valor pode ser PROCESS , DEFAULTED ou SUBSTITUTED . PROCESS - O valor é fornecido pela entrada do processo I/O ou calculado por alguma função. DEFAULTED - O valor contém um valor padrão. SUBSTITUTED - O valor é fornecido pelo operador ou por uma fonte automática.
90	SpillwayGateType	Tipo de portão do vertedouro
91	SurgeTankCode	Tipo (ou a ausência) do <i>surgetank</i> que está associado com a hidrelétrica
92	Susceptance	Parte imaginária da admitância, expressa em Siemens
93	SwitchState	Estados possíveis para uma chave interruptora {open, close}
94	SynchronousMachineOperating Mode	Modalidade de operação da máquina síncrona {generator, condenser}
95	SynchronousMachineType	Tipo da máquina síncrona {generator, condenser, generator_or_condenser}
96	TapStep	Degrau de posicionamento do "tap" do transformador. As posições são numeradas sempre sequencialmente começando com "1" na posição mais baixa, e progredindo até a mais alta posição do "tap"

continua

conclusão

ATRIBUTOS		DEFINIÇÃO
97	Temperature	Valor da temperatura em TemperatureUnits
98	TemperatureUnits	Unidades para a medida de temperatura {Celsius, Fahrenheit}
99	TerminalCount	Número máximo de terminais para um equipamento condutor.
100	TerminalType	Uma designação da função de um terminal em seu equipamento condutor associado.
101	TimeStamp	Data e Hora em "yyy-mm-ddThh:mm:ss", conforme ISO 8601 usando a forma estendida e alguma opção de zona
102	TransformerControlMode	Modo de controle de um transformador {Off, Local, Volt, MW, MVar}
103	TransformerCoolingType	Tipo de refrigeração do transformador
104	TransformerType	Tipo do transformador {fix, voltageControl, phaseControl, voltageAndPhaseControl}
105	TurbineType	Tipo de turbina {francis, pelton, kaplan}
106	Ulong	Longo sem sinal
107	UlongLong	
108	Validity	<p>O valor é GOOD se nenhuma condição anormal da função de aquisição ou da fonte de informação for detectada.</p> <p>O valor é QUESTIONABLE se uma função de supervisão detectar um comportamento anormal, porém o valor poderia ainda ser válido. O cliente é responsável para determinar se "QUESTIONABLE" deve ser usado.</p> <p>O valor é INVALID quando uma função de supervisão reconhece condições anormais da função de aquisição ou da fonte de informação (ausente ou atualização inoperante). O valor não é definido sob esta circunstância. O valor INVALID é usado para indicar ao cliente que o valor pode estar incorreto e não deverá ser usado.</p>
109	Voltage	Valor representando a tensão (kV)
110	Volume	Volume de água do reservatório, dado em milhões de metros cúbicos (Mm3)
111	WaterLevel	O nível de água do reservatório tendo como referência o nível do mar, em metros
112	WindingConnection	Tipo de conexão do enrolamento: delta, estrela ou zigzag {D, Y, Z}
113	WindingType	O tipo de enrolamento {Primário, Secundário, Terciário, Quaternário}
114	XAxisType	O tipo de variável independente no eixo x {absoluteTime, anyRealValue, normalizedDailyTime}
115	XAxisUnits	As unidades de medida do eixo x {absoluteTime, normalizedTime, MW}
116	Y1AxisUnits	As unidades de medida do eixo Y1 {unitOfCurrencyPerHour, MW, MVar, integer}
117	Y2AxisUnits	As unidades de medida do eixo Y2 {MW, MVar, integer}
118	YAxisType	O tipo de variável dependente {singleYValue, twoYValues}

A.1.3. Pacote *Generation*: Diagrama de Classes

O Diagrama de Classes do Pacote *Generation* (main), é apresentado na Figura 26.

Figura 26. Diagrama de Classes do Pacote *Generation* (main) - [6]

A.1.4. Pacote *Generation / GenerationDynamics*: Diagramas de Classes

A Figura 27, apresenta o Diagrama de Classes do Pacote *Generation / GenerationDynamics*.

Figura 27. Diagrama de Classes do Pacote *Generation / GenerationDynamics* - [6]

A.1.4.1. Pacote *Generation / GenerationDynamics*: Classe – BWRSteamSupply

Classe derivada de SteamSupply. Representa o reator de ebulição de água (*Boiling Water Reactor*), usado como fonte de vapor para uma turbina a vapor, para maiores detalhes consulte o ANEXO C, item “C.1”. Os atributos da classe são:

- **highPowerLimit**: indica o limite máximo de potência.
- **inCoreThermalTC**: indica uma constante de tempo (segundos) do núcleo térmico.

- **integralGain**: indica o ganho integral.
- **lowerLimit**: indica o limite mínimo inicial (PU).
- **lowPowerLimit**: indica o limite mínimo de potência (PU).
- **pressureLimit**: indica o limite de pressão (PU).
- **pressureSetpointGA**: ajustador de ganho do ponto de referência para controle (*setpoint*) de pressão.
- **pressureSetpointTC1**: constante de tempo (segundos) do primeiro ponto de controle de pressão.
- **pressureSetpointTC2**: constante de tempo (segundos) do segundo ponto de controle de pressão.
- **proportionalGain**: indica o ganho proporcional.
- **rfAux1**: indica o coeficiente de modelagem, do efeito da falta de frequência e tensão na circulação e fluxo principal do reator de ebulição de água (BWR), o qual afeta a saída (produção) de potência. No total são oito atributos: de **rfAux1** até **rfAux8** (PU).
- **rodPattern**: indica o padrão da vareta (haste utilizada na parada programada do reator), em PU.
- **rodPatternConstant**: constante associada com o padrão da vareta.
- **upperLimit**: indica o limite máximo inicial (PU).

A.1.4.2. Pacote *Generation / GenerationDynamics*: Classe – *CombustionTurbine*

Classe derivada da PrimeMover. Indica o motor primário, que normalmente é abastecido por gás ou óleo. Os atributos da classe são:

- **ambientTemp**: indica a temperatura ambiente padrão para ser usada na modelagem de aplicações.
- **auxPowerVersusFrequency**: indica o efeito da ausência da frequência nominal nas turbinas auxiliares. Por redução da unidade, no consumo MW auxiliar, contra a redução da unidade, na frequência (da frequência nominal).
- **auxPowerVersusVoltage**: indica o efeito da falta de tensão nas turbinas auxiliares. Por redução da unidade, no consumo de MW auxiliar, contra a redução da unidade, na tensão auxiliar do barramento (de um nível de tensão especificado).
- **capabilityVersusFrequency**: indica o efeito da falta de frequência na potência da turbina. Por redução da unidade, em MW de potência, contra a redução da unidade, na frequência (da frequência nominal).
- **heatRecoveryFlag**: é verdadeiro, se a turbina de combustão está associada a uma caldeira de recuperação de calor.
- **powerVariationByTemp**: indica a mudança na unidade, em potência, contra a mudança na unidade, pela temperatura ambiente.
- **referenceTemp**: indica a temperatura de referência a qual a produção da turbina foi definida.

- **timeConstant**: indica a constante de tempo para a turbina.

A.1.4.3. Pacote *Generation / GenerationDynamics*: Classe – CTTempMWCurve

Classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a taxa bruta da saída de potência da turbina à combustão, em MW (eixo X), e a temperatura do ar ambiente, em unidades de temperatura (eixo Y). O atributo da classe é:

- **temperatureUnits**: indica a unidade de temperatura (°C ou °F).

A.1.4.4. Pacote *Generation / GenerationDynamics*: Classe – DrumBoiler

É uma subclasse de FossilSteamSupply. Representa as caldeiras de tambor (*drum boiler*), usadas em condições subcríticas de pressão de vapor. A caldeira de tambor é uma tecnologia comprovada e apropriada para plantas de pequena e média capacidade. O atributo da classe é:

drumBoilerRating: indica a categoria da caldeira de tambor em unidades de vapor.

A.1.4.5. Pacote *Generation / GenerationDynamics*: Classe – FossilSteamSupply

Classe derivada de SteamSupply. Descreve a caldeira que utiliza combustível fóssil, como: carvão, óleo e gás. Os atributos da classe são:

- **auxPowerVersusFrequency**: indica o efeito da ausência da frequência nominal sobre a potência auxiliar real. Por variação do MW contra a variação de frequência, da unidade.
- **auxPowerVersusVoltage**: indica o efeito da falta de tensão sobre a potência auxiliar real. Por variação do MW contra a variação do kV, da unidade.
- **controlIC**: indica uma constante integral.

- **boilerControlMode**: indica a modalidade de controle da caldeira.
- **controlMWEB**: controle de predeterminação (tendência) de erro no MW (*MW Error Bias*).
- **controlPC**: indica uma constante proporcional.
- **controlPEB**: controle de predeterminação (tendência) de erro na pressão (*Pressure Error Bias*).
- **controlPED**: controle da faixa de tolerância de escala (*deadband*), de erro na pressão (*Pressure Error Deadband*).
- **controlTC**: indica uma constante de tempo.
- **feedWaterIG**: indica o ganho integral do alimentador de água.
- **feedWaterPG**: indica o ganho proporcional do alimentador de água.
- **feedWaterTC**: indica uma constante de tempo (segundos) do alimentador de água.
- **fuelDemandLimit**: indica o limite de tempo (segundos) de demanda de combustível.
- **fuelSupplyDelay**: indica o limite de tempo (segundos) na demora de provisão de combustível.
- **mechPowerSensorLag**: indica a desfasagem do sensor de força mecânica.
- **fuelSupplyTC**: indica uma constante de tempo (segundos) de provisão de combustível.
- **mWMMaximumER**: indica o limite máximo da taxa de erro de MW.

- **mWMinimumER**: indica o limite mínimo da taxa de erro de MW.
- **pressureCtrlDG**: indica a proporção do ganho derivado (secundário) do controle de pressão.
- **pressureCtrlIG**: indica a proporção do ganho integral do controle de pressão.
- **pressureCtrlPG**: indica a razão do ganho proporcional do controle de pressão.
- **pressureFeedback**: indicador de avaliação de pressão.
- **superHeater1Capacity**: indica a capacidade do superaquecedor do tambor primário.
- **superHeater2Capacity**: indica a capacidade do superaquecedor do tambor secundário.
- **superHeaterPipePD**: constante de queda de pressão da tubulação do superaquecedor.
- **throttlePressureSP**: indica o valor de referência (*setpoint*) do regulador de pressão.

A.1.4.6. Pacote *Generation / GenerationDynamics*: Classe – HeatRecoveryBoiler

Classe derivada de FossilSteamSupply. Representa o sistema de recuperação de calor associado às turbinas de combustão, no intuito de produzir vapor para as plantas de ciclo combinado. O atributo da classe é:

- **steamSupplyRating2**: indica a taxa de provisão de vapor em kilopounds/hora, se for caldeira dupla de pressão.

A.1.4.7. Pacote *Generation / GenerationDynamics*: Classe – **HydroTurbine**

Classe derivada da PrimeMover. Indica o motor primário movido a água. Os tipos de turbinas são: *Francis*, *Kaplan* e *Pelton*. Os atributos da classe são:

- **gateRateLimit**: indica o limite padrão da comporta.
- **gateUpperLimit**: indica o limite superior da comporta.
- **maxEffMWatMinHead**: indica a eficiência máxima de MW em condições de mínima “head”.
- **maxEffMWatMaxHead**: indica a eficiência máxima de MW em condições de máximas “head”.
- **speedRating**: indica a velocidade calculada em revoluções por minuto (rpm).
- **speedRegulation**: indica a velocidade nominal (padrão).
- **transientDroopTime**: indica a constante de tempo (segundos) de queda ou enfraquecimento passageiro.
- **transientRegulation**: indica um ajuste transiente (passageiro).
- **turbineRating**: indica a potência estimada da turbina em MW.
- **turbineType**: indica o tipo de turbina, por exemplo: *Francis*, *Pelton* e *Kaplan*.
- **waterStartingTime**: indica o tempo (segundos) do início da ação da água.

A.1.4.8. Pacote *Generation / GenerationDynamics*: Classe – PrimeMover

Classe derivada de PowerSystemResource (Pacote *Core*). É a máquina utilizada para desenvolver energia mecânica, usada para impulsionar um gerador. O atributo da classe é:

- **primeMoverRating**: indica a categoria (classificação) do motor primário.

A.1.4.9. Pacote *Generation / GenerationDynamics*: Classe – PWRSteamSupply

Classe derivada de SteamSupply. Representa o reator de água pressurizada (*Pressurized Water Reactor*), usado como fonte de vapor para uma turbina a vapor, para maiores detalhes consulte o ANEXO C, item “C.2”. Os atributos da classe são:

- **coldLegFBLagTC**: constante de tempo de atraso de realimentação do trecho (percurso) frio (PU).
- **coldLegFBLeadTC1**: constante 1 de tempo de ligação da realimentação do trecho frio (PU).
- **coldLegFBLeadTC2**: constante 2 de tempo de ligação da realimentação do trecho frio (PU).
- **coldLegFG1**: ganho 1 na realimentação do trecho frio (PU).
- **coldLegFG2**: ganho 2 na realimentação do trecho frio (PU).
- **coldLegLagTC**: constante de tempo de atraso do trecho frio (PU).
- **coreHTLagTC1**: constante 1 de tempo de atraso de transferência de calor do núcleo (PU).

- **coreHTLagTC2**: constante 2 de tempo de atraso de transferência de calor do núcleo (PU).
- **coreNeutronicsEffTC**: constante de tempo efetivo de neutrônicos⁷ do núcleo (PU).
- **coreNeutronicsHT**: indica a transferência de calor e neutrônicos do núcleo (PU).
- **feedbackFactor**: fator de regeneração (PU).
- **hotLegLagTC**: constante de tempo de atraso do trecho quente (PU).
- **hotLegSteamGain**: indica o ganho de vapor do trecho quente (PU).
- **hotLegToColdLegGain**: indica o ganho do trecho quente para o trecho frio (PU).
- **pressureCG**: indica o ganho do controle de pressão (PU).
- **steamFlowFG**: indica o ganho da regeneração do fluxo de vapor (PU).
- **steamPressureDropLagTC**: constante de tempo no atraso de queda da pressão do vapor (PU).
- **steamPressureFG**: indica o ganho da regeneração da pressão do vapor (PU).
- **throttlePressureFactor**: indica o fator de regulagem de pressão (PU).
- **throttlePressureSP**: indica o valor de referência (*setpoint*) do regulador de pressão (PU).

⁷ Neutrônicos, estudo do movimento dos nêutrons em materiais fragmentáveis ou não, e das reações que induzem na matéria, em particular dentro dos reatores nucleares.

A.1.4.10. Pacote *Generation / GenerationDynamics*: Classe – **SteamSupply**

Classe derivada de PowerSystemResource (Pacote *Core*). Indica a provisão de vapor para a turbina a vapor. O atributo da classe é:

- **steamSupplyRating**: indica a categoria (classificação) da fonte de vapor.

A.1.4.11. Pacote *Generation / GenerationDynamics*: Classe – **SteamTurbine**

É uma subclasse de PrimeMover. Representa uma turbina a vapor. Os atributos da classe:

- **crossoverTC**: indica a constante de tempo (segundos) de contato (conexão).
- **reheater1TC**: indica a primeira constante de tempo (segundos) de reaquecimento.
- **reheater2TC**: indica a segunda constante de tempo (segundos) de reaquecimento.
- **shaft1PowerHP**: indica a fração de potência da saída de alta pressão da turbina, do eixo 1 (espécie de mecanismo giratório para geração de energia).
- **shaft1PowerIP**: indica a fração de potência da saída intermediária de pressão da turbina, do eixo 1.
- **shaft1PowerLP1**: indica a fração de potência da primeira saída de baixa pressão da turbina, do eixo 1.
- **shaft1PowerLP2**: indica a fração de potência da segunda saída de baixa pressão da turbina, do eixo 1.
- **shaft2PowerHP**: indica a fração de potência da saída de alta pressão da turbina, do eixo 2.

- **shaft2PowerIP**: indica a fração de potência da saída intermediária de pressão da turbina, do eixo 2.
- **shaft2PowerLP1**: indica a fração de potência da primeira saída de baixa pressão da turbina, do eixo 2.
- **shaft2PowerLP2**: indica a fração de potência da segunda saída de baixa pressão da turbina, do eixo 2.
- **steamChestTC**: indica a constante de tempo (segundos) da caldeira.

A.1.4.12. Pacote *Generation / GenerationDynamics*: Classe – Subcritical

É uma subclasse de FossilSteamSupply. Representa a caldeira de passagem de pressão subcrítica (*once-through subcritical boiler*). Os atributos são os mesmos especificados na superclasse.

A.1.4.13. Pacote *Generation / GenerationDynamics*: Classe – Supercritical

É uma subclasse de FossilSteamSupply. Representa a caldeira de passagem de pressão supercrítica⁸ (*once-through supercritical boiler*). Este tipo de caldeira opera geralmente acima da pressão crítica, sem haver nenhuma circulação do líquido de funcionamento em qualquer parte da unidade. Os atributos são os mesmos especificados na superclasse.

⁸ Supercrítica, é uma expressão da termodinâmica que descreve o estado de uma substância onde não existe distinção entre o líquido e a fase gasosa (isto é, são fluidos homogêneos). A água alcança este estado a uma pressão acima de 22.1 megapascals (MPa).

A.1.5. Pacote *Generation / Production*: Diagrama de Classes

A Figura 28, apresenta o Diagrama de Classes do Pacote *Generation / Production* (main).

Figura 28. Diagrama de Classes do Pacote *Generation / Production* (main) - [6]

É uma subclasse de CurveSchedule (Pacote *Core*), sua finalidade e atributos são os mesmos especificados na superclasse.

A.1.5.2. Pacote *Generation / Production*: Classe – AirCompressor

Classe derivada de PowerSystemResource (Pacote *Core*). Representa o compressor de ar da turbina de combustão que é parte integrante de uma planta CAES (*Compressed Air Storage Energy*). O atributo da classe é:

- **airCompressorRating**: classificação do compressor de ar da planta de CAES.

A.1.5.3. Pacote *Generation / Production*: Classe – CAESPlant

Classe derivada de PowerSystemResource (Pacote *Core*). Representa a planta de armazenamento de energia de ar comprimido (CAES – *Compressed Air Storage Energy*). Os atributos da classe são:

- **energyStorageCapacity**: é o cálculo da capacidade de armazenamento de energia em megawatt-horas (MWh).
- **ratedCapacityMW**: indica a capacidade geradora nominal bruta da planta CAES em MW.

A.1.5.4. Pacote *Generation / Production*: Classe – CogenerationPlant

Classe derivada de PowerSystemResource (Pacote *Core*). Representa um conjunto de unidades geradoras térmicas para a produção de energia elétrica e emissão de vapor (saída das turbinas de vapor). A emissão do vapor é usada tipicamente para finalidades industriais ou para o aquecimento e refrigeração municipais. Os atributos da classe são:

- **cogenHPSendoutRating**: indica a emissão de vapor de alta pressão, em klb/hora.

- **cogenHPSteamRating**: indica a taxa de vapor de alta pressão, em libras por polegada quadrada (psi).
- **cogenLPSteamRating**: indica a taxa de vapor de baixa pressão, em libras por polegada quadrada (psi).
- **cogenLPSteamRating**: indica a taxa de vapor de baixa pressão, em libras por polegada quadrada (psi).
- **cogenPlantMWRating**: indica a saída calculada de MW da planta de cogeração.

A.1.5.5. Pacote *Generation / Production*: Classe – CombinedCyclePlant

Classe derivada de PowerSystemResource (Pacote *Core*). Representa um conjunto de turbinas de combustão e de turbinas a vapor, onde o calor da exaustão das turbinas da combustão é recuperado para fazer o vapor para as turbinas a vapor, tendo por resultado uma maior eficiência global da planta. O atributo da classe é:

- **combCyclePlantRating**: é o cálculo de saída da planta de ciclo combinado, em MW.

A.1.5.6. Pacote *Generation / Production*: Classe – EmissionAccount

É uma classe derivada de AccountBalance. São contas para localizar uso e saldo de emissões para as unidades geradoras térmicas. Uma unidade pode ter nenhuma ou várias contas de emissão, e normalmente terá uma para localizar o uso, e outra para localizar os saldos de emissões. Os atributos da classe são:

- **emissionType**: indica o tipo de emissão.
- **emissionValueSource**: é o valor da fonte de emissão, que pode ser medido ou calculado.

A.1.5.7. Pacote *Generation / Production*: Classe – **EmissionCurve**

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a taxa de emissão da unidade, em unidades de massa por hora (eixo Y), e a saída em MW (eixo X), para um determinado tipo de poluente. Esta curva aplica-se somente quando um único tipo de combustível está sendo queimado. Os atributos da classe são:

- **emissionContent**: é o índice de emissão por quantidade de combustível queimado.
- **emissionType**: indica o tipo de emissão, além da taxa de produção da unidade medida. O “y1AxisUnits” da curva, contém a unidade de medida (exemplo: Kg), e o tipo de emissão (emissionType) seria, por exemplo, o dióxido de enxofre.
- **netGrossMWFlag**: quando o valor é verdadeiro, significa que a produção está expressa em MW líquido.

A.1.5.8. Pacote *Generation / Production*: Classe – **FossilFuel**

É uma classe derivada de Naming (Pacote *Core*). Indica o combustível fóssil consumido pela unidade geradora térmica (não nuclear), por exemplo: gás, óleo e carvão. Os atributos da classe são:

- **fossilFuelType**: indica o tipo de combustível fóssil, tais como: gás, óleo e carvão.
- **fuelCost**: o custo referente ao valor térmico para o determinado tipo de combustível.
- **fuelDispatchCost**: indica o custo do combustível para o despacho econômico, o qual inclui: custo do combustível, custo do transporte e o custo de manutenção adicional.

- **fuelEffFactor**: indica o fator de eficiência do combustível (por unidade) em termos de MBtu real absorvido.
- **fuelHandlingCost**: indica o custo de manipulação e processamento do combustível.
- **fuelHeatContent**: indica a quantidade de calor (Btu) por peso (ou volume) do determinado tipo de combustível.
- **fuelMixture**: indica a quantidade em percentagem do determinado tipo de combustível, quando diversos combustíveis são consumidos.
- **fuelSulfur**: indica a fração do combustível de crédito de poluição por unidade de índice de calor (MBtu).
- **highMWBreakpoint**: indica o MW produzido pela unidade geradora em que o determinado tipo de combustível está ligado. Este combustível (por exemplo, óleo) é utilizado às vezes para suplementar o combustível básico (por exemplo, carvão) na produção máxima de MW.
- **lowMWBreakpoint**: indica o MW produzido pela unidade geradora em que o determinado combustível está desligado. Este combustível (por exemplo, óleo) é utilizado às vezes para suplementar o combustível base (por exemplo, carvão) na produção mínima de MW.

A.1.5.9. Pacote *Generation / Production*: Classe – FuelAllocationSchedule

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a quantidade de combustível de um determinado tipo que é utilizado para consumo durante um período de tempo especificado. Os atributos da classe são:

- **fuelAllocationEndDate**: indica a data e hora programada de término de utilização do combustível.
- **fuelAllocationStartDate**: indica a data e hora programada de início de utilização do combustível.
- **fuelType**: indica o tipo de combustível e a correspondente unidade de medida.
- **maxFuelAllocation**: indica a quantidade máxima de combustível que será utilizada para consumo no período de tempo programado.
- **minFuelAllocation**: indica a quantidade mínima de combustível que será utilizada para consumo no período de tempo programado.

A.1.5.10. Pacote *Generation / Production*: Classe – GeneratingUnit

É uma subclasse de Equipment, e representa uma única máquina, ou um conjunto de máquinas síncronas para converter força mecânica em energia elétrica (corrente alternada). Por exemplo, uma máquina de um conjunto pode estar definida para utilização em horários preestabelecidos quando um sinal de controle for emitido ao conjunto das máquinas. Neste caso, haverá uma GeneratingUnit para cada membro deste conjunto de máquinas e uma GeneratingUnit adicional correspondente ao conjunto. Os atributos da classe são:

- **ControlDeadband**: unidade de controle de erro na faixa de tolerância de escala (deadband). Quando a mudança de MW de uma unidade for menor que esta zona morta, significa que nenhum pulso de controle será enviado à unidade. A zona morta, pode ser um valor fixo ou ajustável.
- **controlPulseHigh**: limite máximo de pulsos de controle, é a maior quantidade de pulsos de controle que a unidade pode responder.
- **controlPulseLow**: limite mínimo de pulsos de controle, é a menor quantidade de pulsos de controle que a unidade pode responder.
- **controlResponseRate**: taxa de resposta da unidade, especifica a mudança de MW para um pulso do controle de um segundo, no mais suscetível nível de carga da unidade.
- **efficiency**: indica a eficiência da unidade em converter energia mecânica, do motor principal, em energia elétrica (medida em PU).
- **genControlMode**: indica a seleção da modalidade de controle da unidade como *setpoint* (S) ou pulso (P).
- **genControlSource**: indica a forma de controle da unidade geradora, isto é: *Unavailable* (não disponível), *Off-AGC* (não faz parte do CAG), *On-AGC* (faz parte do CAG) e *Plant Control* (controle da planta).
- **governorMPL**: indica o limite de posição do governador do gerador (MPL – *Motor Position Limit*).

- **governorSCD**: indica a inclinação do comutador de velocidade do regulador (SCD – *Speed Changer Droop*).
- **highControlLimit**: indica o limite máximo para o controle secundário (CAG).
- **initialMW**: valor inicial de MW, que é usado para armazenar um resultado do fluxo de potência (powerflow) para o MW inicial da unidade, numa determinada configuração da rede.
- **lowControlLimit**: indica o limite mínimo para o controle secundário (CAG).
- **maximumAllowableSpinningReserve**: reserva de giro máximo permitida. A reserva de giro nunca será considerada maior que este valor, indiferentemente do ponto de operação corrente.
- **maximumEconomicMW**: limite máximo de MW econômico, que não deve exceder o limite máximo de MW operacional.
- **maximumOperatingMW**: limite máximo de MW operacional que pode ser estabelecido para unidade geradora.
- **minimumEconomicMW**: limite mínimo de MW econômico, que deve ser maior ou igual ao limite mínimo de MW operacional.
- **minimumOperatingMW**: limite mínimo de MW operacional que pode ser estabelecido para a unidade geradora.
- **modelDetail**: nível de detalhe dos dados do modelo de gerador.

- **ratedGrossMaxMW**: capacidade máxima nominal bruta da unidade (especificação do fabricante).
- **ratedGrossMinMW**: nível de geração mínima nominal bruta que a unidade pode operar seguramente, durante a entrega de potência à rede de transmissão.
- **ratedNetMaxMW**: capacidade máxima nominal líquida, determinada pela subtração da potência auxiliar utilizada para operar a maquinaria da usina, da capacidade máxima bruta calculada.
- **startupTime**: intervalo de tempo para colocar a unidade “on-line”, a partir do instante que a força mecânica for aplicada.
- **autoCntrlMarginMW**: capacidade não utilizada de planejamento, que pode ser usada para suportar o controle automático de sobrecarga (overruns), em MW.
- **allocSpinResMW**: capacidade não utilizada de planejamento (reserva de giro), que pode ser usada para suportar uma carga de emergência.
- **baseMW**: para as unidade que permitem variação de carga, este valor representa o ponto base do MW econômico, para as unidade que não permite a variação de carga, este é o valor fixo de geração. O valor deve estar entre os limites mínimo e máximo de operação.
- **dispReserveFlag**: indicador booleano de *Reserve*
- **energyMinMW**: calor gerado, em milhões de Btus, por hora do tempo decorrido (MBtu/Hora).
- **fastStartFlag**: indicador booleano de partida rápida (*fast start*)

- **FuelPriority**: define o nível de prioridade de abastecimento com combustível. Zero significa prioridade máxima.
- **genOperatingMode**: modo operacional do controle secundário, valores possíveis: “Off”, “Manual”, “Fixed”, “LFC”, “AGC”, “EDC”, “RPN”, “MRN” ou “REG”.
- **longPF**: fator de participação econômica (longa) da unidade geradora.
- **lowerRampRate**: taxa de mudança (redução) do Megawatt por minuto.
- **normalPF**: fator de participação econômica (normal) da unidade geradora.
- **penaltyFactor**: definido como, “ $1 / (1 - \text{Perda de Transmissão Incremental})$ ”; a Perda de Transmissão Incremental pode ser expressa por valores positivos ou negativos. O intervalo típico do Fator de Penalidade é de (0.9 a 1.1).
- **raiseRampRate**: taxa de mudança (aumento) do Megawatt por minuto.
- **shortPF**: fator de participação econômica (curta) da unidade geradora.
- **spinReserveRamp**: taxa de rampa de giro de reserva do recurso, em MW/minuto.
- **stepChange**: produto do valor RMS da tensão e do valor RMS do componente da corrente em fase (Megawatt - MW).
- **tieLinePF**: fator de participação econômica da unidade geradora.

A.1.5.11. Pacote *Generation / Production*: Classe – GenUnitOpCostCurve

É uma subclasse de CurveSchedule (Pacote *Core*). Indica o relacionamento entre o custo operacional da unidade em \$/hora (eixo Y) e a saída da unidade em MW (eixo X). A curva do

custo operacional para as unidades térmicas é derivada da intensidade do calor e dos custos de combustível. A curva do custo operacional para as unidades hídricas é derivada das taxas de fluxo da água e dos custos equivalentes da água. O atributo da classe é:

- **netGrossMWFlag**: quando o valor é verdadeiro, significa que a produção está expressa em MW líquido.

A.1.5.12. Pacote *Generation / Production*: Classe – GenUnitOpSchedule

Classe derivada de CurveSchedule (Pacote *Core*). Informa a programação operacional atual (ou plano) aprovada pelo operador da unidade geradora, produzida tipicamente com o auxílio das análises do tipo de compromisso da unidade. O eixo X representa o tempo absoluto. O eixo Y1 representa o estado (0=“off-line” e não disponível; 1=disponível; 2=deve funcionar; 3=deve funcionar em valor fixo de MW; etc.). O eixo Y2 representa o valor fixo de MW de funcionamento onde requerido.

A.1.5.13. Pacote *Generation / Production*: Classe – GrossToNetMWCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica o relacionamento entre a saída bruta de MW da unidade geradora no eixo X (medida nos terminais da(s) máquina(s)) e a saída líquida de MW da unidade geradora no eixo Y (baseada nas medidas definidas na estação de potência).

A.1.5.14. Pacote *Generation / Production*: Classe – HeatInputCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre o fornecimento de calor para a unidade em MBtu por hora, do combustível principal (eixo Y1) e o combustível suplementar (eixo Y2), contra a produção em MW da unidade (eixo X). A

quantidade de combustível principal é utilizada para sustentar a geração num determinado nível de produção, é distribuída proporcionalmente entre os pontos de definição. A quantidade de combustível suplementar utilizado neste nível de produção é fixo e não distribuído proporcionalmente. Os atributos da classe são:

- **auxPowerOffset**: potência de saída – fator de ajuste de compensação (*offset*) da potência auxiliar em MW.
- **auxPowerMult**: potência de saída – fator de ajuste multiplicador da potência auxiliar por unidade.
- **heatInputEff**: calor de entrada – fator de ajuste multiplicador de eficiência por unidade.
- **heatInputOffset**: calor de entrada – fator de ajuste de compensação (*offset*) em MBtu/hora.

A.1.5.15. Pacote *Generation / Production*: Classe – HeatRateCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a taxa de calor da unidade em MBtu/hora por MW (eixo Y) e a produção da unidade em MW (eixo X). A entrada de calor é de todos os combustíveis. O atributo da classe é:

- **netGrossMWFlag**: quando o valor é verdadeiro, significa que a produção está expressa em MW líquido.

A.1.5.16. Pacote *Generation / Production*: Classe – HydroGeneratingEfficiencyCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a eficiência da unidade geradora em percentagem e a saída da unidade em MW para um determinado “head”

líquido em metros. A relação entre a eficiência, descarga, “head” e potência de saída é expressa pela seguinte fórmula: $E = KP/HQ$; onde: ($E = \%$), ($P = \text{MW}$), ($H = \text{m}$), ($Q = \text{m}^3/\text{segundo}$) e ($K = \text{constante}$). Por exemplo, um ponto da curva para um determinado “head” líquido poderia relacionar a eficiência (eixo Y) contra a saída em MW ou contra a descarga sobre o eixo X.

A.1.5.17. Pacote *Generation / Production*: Classe – HydroGeneratingUnit

É uma classe derivada de GeneratingUnit. Representa uma unidade geradora cuja fonte de energia mecânica é uma turbina hidráulica (por exemplo: *Francis, Pelton, Kaplan*). O atributo da classe é:

- **hydroUnitWaterCost**: indica o custo equivalente da água que aciona a turbina hidráulica, expressa em (\$/hora) por ($\text{m}^3/\text{segundo}$).

A.1.5.18. Pacote *Generation / Production*: Classe – HydroPowerPlant

É uma classe derivada de PowerSystemResource (Pacote *Core*). Representa uma central elétrica que utiliza recursos hídricos para geração ou bombeamento. Na geração, as turbinas do gerador recebem a água de um reservatório superior. No bombeamento, as bombas recebem a água de um reservatório mais baixo. Os atributos da classe são:

- **dischargeTravelDelay**: indica o tempo que a água demora para viajar de uma central hidrelétrica até outra, localizada a jusante.
- **hydroPlantType**: indica o tipo de hidrelétrica, por exemplo: fio d’água (*run-of-river*), armazenamento bombeado (*pumped storage*), armazenamento principal (*major storage*) e armazenamento secundário (*minor storage*).

- **penstockType**: indica o tipo e configuração da(s) comporta(s) da hidrelétrica.
- **plantDischargeCapacity**: indica a capacidade de descarga total da hidrelétrica em metros cúbicos por segundo.
- **plantMWGenRating**: é a potência nominal de geração da hidrelétrica em MW calculada para condições de ponta (pico de utilização).
- **plantMWPumpRating**: é a potência nominal de bombeamento em MW calculada para condições de ponta (pico de utilização).
- **plantRatedHead**: nas usinas hidrelétricas a água deve cair do ponto mais alto de sua elevação (*forebay* – nível máximo do reservatório) até o ponto mais baixo (*tailbay* – nível de liberação de água da represa), para liberar a energia armazenada. O atributo indica o “head”, que é a diferença em metros entre estes dois pontos (*forebay* e *tailbay*).
- **surgeTankCode**: é um código que descreve o tipo (ou a ausência) do *surgetank* que está associado com a hidrelétrica.
- **surgeTankCrestLevel**: indica o nível máximo do vertedouro.

A.1.5.19. Pacote *Generation / Production*: Classe – HydroPump

Classe derivada de PowerSystemResource (Pacote *Core*). Representa uma bomba movida a motor síncrono, tipicamente associada à uma planta de armazenamento bombeado. Os atributos da classe são:

- **pumpDischAtMaxHead**: indica a descarga de bombeamento (m^3/seg) sob condições de pressão máxima, normalmente em comporta totalmente aberta.

- **pumpDischAtMinHead**: indica a descarga de bombeamento (m^3/seg) sob condições de pressão mínima, normalmente em comporta totalmente aberta.
- **pumpPowerAtMaxHead**: indica a potência de bombeamento (MW) sob condições de pressão máxima, normalmente em comporta totalmente aberta.
- **pumpPowerAtMinHead**: indica a potência de bombeamento (MW) sob condições de pressão mínima, normalmente em comporta totalmente aberta.

A.1.5.20. Pacote *Generation / Production*: Classe – HydroPumpOpSchedule

Classe derivada de CurveSchedule (Pacote *Core*). Representa a programação operacional atual (ou plano) aprovada pelo operador da bomba hidráulica, produzida com o auxílio das análises de tipo de compromisso da unidade. O estado (status) de programação operacional da unidade é fornecido da seguinte forma: (0 = não disponível), (1 = disponível para iniciar ou parar) e (2 = deve bombear).

A.1.5.21. Pacote *Generation / Production*: Classe – IncrementalHeatRateCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a taxa de calor incremental da unidade em (delta MBtu/hora) por (delta MW), e a saída da unidade em MW. A curva IHR (Incremental Heat Rate), representa a inclinação da HeatInputCurve. Note que a “incremental heat rate” e a “heat rate” possuem a mesma unidade de engenharia, isto é, MBtu/MWh.

A.1.5.22. Pacote *Generation / Production*: Classe – InflowForecast

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica o fluxo de água natural para um reservatório, estimativa de chuva e de degelo. Medida em incrementos de uma hora por

até 10 dias. A previsão é dada em média de metros cúbicos por segundo sobre o incremento de tempo.

A.1.5.23. Pacote *Generation / Production*: Classe – LevelVsVolumeCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre o volume do reservatório em milhões de metros cúbicos e o nível do reservatório em metros.

A.1.5.24. Pacote *Generation / Production*: Classe – PenstockLossCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a perda do “head” (em metros) e o total da descarga através da comporta (em metros cúbicos por segundo). Uma ou mais turbinas podem estar conectadas a mesma comporta.

A.1.5.25. Pacote *Generation / Production*: Classe – Reservoir

É uma subclasse de PowerSystemResource (Pacote *Core*). Representa a facilidade de armazenamento de água dentro de um sistema hídrico, incluindo lagoas, lagos, lagoas e rios. O armazenamento normalmente se dá em algum tipo de represa. Os atributos da classe são:

- **activeStorageCapacity**: indica o volume de armazenamento em Mm^3 , entre o nível de provisão cheio e o nível mínimo operacional normal.
- **energyStorageRating**: indica o armazenamento de energia do reservatório em MWh, calculado para condições de ponta (pico de utilização).
- **fullSupplyLevel**: indica o nível máximo de provisão, sobre o qual a água transbordará do reservatório. Este pode ser considerado o nível de “spillway”, o limite máximo com os vertedouros fechados.

- **grossCapacity**: indica a capacidade total do reservatório, em milhões de metros cúbicos.
- **normalMinOperateLevel**: indica o nível mínimo normal de operação, abaixo do qual as comportas sugarão ar.
- **riverOutletWorks**: saída do rio de trabalho para liberações de direitos ribeirinhos ou outros propósitos.
- **spillTravelDelay**: indica o tempo que a água demora do vertedouro até o próximo reservatório a jusante.
- **spillwayCapacity**: indica a capacidade de fluxo do vertedouro, em metros cúbicos por segundo.
- **spillwayCrestLength**: o comprimento da crista do vertedouro, em metros.
- **spillwayCrestLevel**: nível máximo de água que o vertedouro suporta sem transbordar.
- **spillwayGateType**: tipo de comporta do vertedouro.

A.1.5.26. Pacote *Generation / Production*: Classe – ShutdownCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a taxa bruta de MW/minuto (eixo Y) na qual a unidade deve ser parada e sua saída bruta do momento em MW (eixo X). Os atributos da classe são:

- **shutdownCost**: custo fixo da parada programada.
- **shutdownDate**: data e hora da mais recente parada da unidade geradora.

A.1.5.27. Pacote *Generation / Production*: Classe – StartIgnFuelCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a quantidade de combustível de ignição (eixo Y) usado para reiniciar e compensar a potência auxiliar consumida, contra o número de horas (eixo X) que a unidade esteve desligada (*offline*). O atributo da classe é:

- **ignitionFuelType**: indica tipo de combustível de ignição.

A.1.5.28. Pacote *Generation / Production*: Classe – StartMainFuelCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a quantidade de combustível principal (eixo Y) usado para reiniciar e compensar a potência auxiliar consumida, contra o número de horas (eixo X) que a unidade esteve desligada (*offline*). O atributo da classe é:

- **mainFuelType**: indica o tipo de combustível principal.

A.1.5.29. Pacote *Generation / Production*: Classe – StartRampCurve

É uma classe derivada de Naming (Pacote *Core*). Representa a taxa bruta em MW/minuto (eixo Y) a qual a unidade pode pegar carga, contra o número de horas (eixo X) que a unidade estava desligada (*offline*). O atributo da classe é:

- **hotStandbyRamp**: indica a taxa de inclinação (curva) de partida em MW/minuto bruto para a unidade que está em “hot standby” (prontidão operacional).

A.1.5.30. Pacote *Generation / Production*: Classe – StartupModel

É uma classe derivada de Naming (Pacote *Core*). Indica as características de partida, que depende do tempo que a unidade ficou parada. Os atributos da classe são:

- **fixedMaintCost**: indica o custo fixo de manutenção por hora.
- **hotStandbyHeat**: indica a quantidade de calor requerida por hora para operação em “hot standby” (prontidão operacional).
- **incrementalMaintCost**: indica o custo de manutenção incremental.
- **minimumDownTime**: indica o tempo mínimo, em horas, que a unidade deve permanecer parada, antes de ser ligada (restart).
- **minimumRunTime**: indica o tempo mínimo, em horas, que a unidade deve operar, antes da permissão de parada (*shutdown*).
- **riskFactorCost**: indica o custo da oportunidade associada ao retorno em dinheiro (dólares). Isto representa a “quota” do “restart” (reinício), na depreciação e no risco de ocorrência de um evento que danifique a unidade geradora.
- **startupCost**: indica os custos de partida (*startup*) de uma unidade geradora.
- **startupDate**: indica a data e hora da mais recente partida (*startup*) de uma unidade geradora.
- **startupPriority**: indica a prioridade de partida de uma unidade geradora dentro de uma área de controle, quanto menor o valor, maior é a prioridade. Pode ser atribuída a mesma prioridade para mais de uma unidade em uma área.
- **stbyAuxPowerMW**: indica o consumo de potência da unidade auxiliar para ser mantida em modo “standby” (prontidão).

A.1.5.31. Pacote *Generation / Production*: Classe – SteamSendoutSchedule

Classe derivada de CurveSchedule (Pacote *Core*). Indica a programação de emissão de vapor da planta de cogeração em klb/hora.

A.1.5.32. Pacote *Generation / Production*: Classe – TailbayLossCurve

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica a relação entre a perda do “head” (diferença entre o *forebay* e o *tailbay* em metros no eixo Y), e a descarga total da hidrelétrica no “tailbay” (em m³/segundo no eixo X). Pode haver mais de uma curva dependendo do nível do “tailbay” do reservatório ou do nível do rio.

A.1.5.33. Pacote *Generation / Production*: Classe – TargetLevelSchedule

É uma classe derivada de CurveSchedule (Pacote *Core*). Indica o nível de água do reservatório, com objetivo de estudos avançados ou “curvas de controle”. Os atributos da classe são:

- **highLevelLimit**: nível máximo do reservatório, acima do qual sua operação será penalizada.
- **LowLevelLimit**: nível mínimo do reservatório, abaixo do qual sua operação será penalizada.

A.1.5.34. Pacote *Generation / Production*: Classe – ThermalGeneratingUnit

É uma subclasse de GeneratingUnit. Descreve unidade geradora cuja a máquina principal poderia ser uma turbina a vapor, uma turbina de combustão ou um motor a diesel. O atributo da classe é:

- **oMCost**: indica o custo de operação e manutenção da unidade térmica.

A.1.6. Pacote *LoadModel*: Diagrama de Classes

A Figura 31, apresenta o Diagrama de Classes do pacote *LoadModel*.

Figura 31. Diagrama de Classes do Pacote *LoadModel* - [6]

A.1.6.1. Pacote *LoadModel*: Classe – *AreaLoadCurve*

Classe derivada de *CurveSchedule* (Pacote *Core*). Representa uma curva que relaciona potência contra o tempo, mostrando os valores de uma carga específica de cada unidade no período coberto. A curva pode ser baseada na hora “absoluta” ou “normalizada”. Uma ocorrência desta curva, poderia representar a previsão absoluta de carga da área. Uma outra ocorrência, poderia representar a curva de carga diária normalizada para um particular tipo de dia.

A.1.6.2. Pacote *LoadModel*: Classe – *AreaLossCurve*

Classe derivada de *CurveSchedule* (Pacote *Core*). Representa a relação entre o total da perda de MW (eixo Y) e o total da carga de MW da área (eixo X).

A.1.6.3. Pacote *LoadModel*: Classe – *CustomerMeter*

Classe derivada de *EnergyConsumer* (Pacote *Wires*). Representa o medidor de consumo de energia do cliente. O atributo **typeName** indica o tipo de medidor do cliente.

A.1.6.4. Pacote *LoadModel*: Classe – *DayType*

Classe derivada de *Naming* (Pacote *Core*). Representa o grupo de dias similares, por exemplo, os dias úteis, os fins de semana e os feriados.

A.1.6.5. Pacote *LoadModel*: Classe – *EquivalentLoad*

Classe derivada de *EnergyConsumer* (Pacote *Wires*). Representa o equivalente genérico de um consumidor de energia nas redes de transmissão ou no nível de tensão de distribuição. Os atributos da classe são:

- **feederLoadMgtFactor**: indica a percentagem de contribuição do alimentador para o gerenciamento da carga.
- **mVArColdPickUpFactor**: indica em percentagem a quantia de MVar nominal do alimentador que é selecionado a frio.
- **mWColdPickUpFactor**: indica em percentagem a quantia de MW nominal do alimentador que é selecionado a frio.

- **phaseAmpRating**: indica o valor nominal de corrente em Ampères de uma única fase.
- **loadAllocationFactor**: permite a atribuição das cargas com base no fator de participação, por exemplo: dada três cargas com fatores 10, 25 e 15, então uma carga de 100 Ampères do alimentador poderia ser repartida no alimentador em 20, 50 e 30 Ampères respectivamente.

A.1.6.6. Pacote *LoadModel*: Classe – InductionMotorLoad

Classe derivada de EnergyConsumer (Pacote *Wires*). Representa uma grande carga de um motor de indução trifásico. O atributo **typeName** indica o tipo do motor de indução: 1 = *wound rotor*, 2 = *squirrel cage* (motor de gaiola).

A.1.6.7. Pacote *LoadModel*: Classe – LoadArea

Classe derivada de PowerSystemResource (Pacote *Core*). Representa os grupos de carga, os consumidores de energia.

A.1.6.8. Pacote *LoadModel*: Classe – LoadDemandModel

Classe derivada de CurveSchedule (Pacote *Core*). Representa uma curva de carga versus o tempo (eixo X), mostrando os valores de MW (eixo Y1) e MVar (eixo Y2) para cada unidade de tempo do período coberto. Esta curva representa o padrão típico da carga ao longo do período de tempo para um determinado tipo de dia ou estação climática.

A.1.6.9. Pacote *LoadModel*: Classe – *NonConformLoadSchedule*

Classe derivada de CurveSchedule (Pacote *Core*). Representa as curvas de programação de MW (eixo Y1) e de MVar (eixo Y2) ao longo do tempo (eixo X), para as cargas fora de conformidade, por exemplo, uma grande carga industrial ou de serviço da estação de potência.

A.1.6.10. Pacote *LoadModel*: Classe – *PowerCutZone*

Classe derivada de PowerSystemResource (Pacote *Core*). Representa uma área ou região do sistema de potência a qual é utilizada para propósitos de rejeição de carga (load shedding) em consumidores pré-selecionados de um sistema energético, quando da ocorrência de uma condição anormal, em um esforço para manter a integridade do sistema. Os atributos da classe são:

- **CutLevel1**: indica o primeiro nível (quantidade) de carga a ser rejeitada, como uma percentagem da carga total da região.
- **cutLevel2**: indica o segundo nível (quantidade) de carga a ser rejeitada, como uma percentagem da carga total da região.

A.1.6.11. Pacote *LoadModel*: Classe – *Season*

Classe derivada de Naming (Pacote *Core*). Representa uma específica estação climática: primavera, verão, outono e inverno. Os atributos da classe são:

- **endDate**: indica o data e hora de término da estação.
- **startDate**: indica o data e hora de início da estação.

A.1.6.12. Pacote *LoadModel*: Classe – StationSupply

Classe derivada de EnergyConsumer (Pacote *Wires*). Representa uma estação de fornecimento.

A.1.7. Pacote *Meas*: Diagrama de Classes

O Diagrama de Classes do Pacote *Meas* (main), é apresentado na Figura 32.

Figura 32. Diagrama de Classes do Pacote *Meas* (main) - [6]

A Figura 33, mostra o Diagrama de Classes do Pacote *Meas* (measurements).

Figura 33. Diagrama de Classes do Pacote *Meas* (measurements) - [6]

A Figura 34, apresenta o Diagrama de Classes do Pacote *Meas* (quality).

Figura 34. Diagrama de Classes do Pacote *Meas* (quality) - [6]

A.1.7.1. Pacote *Meas*: Classe – Control

Subclasse de Naming (Pacote *Core*). A classe **Control** é utilizada para supervisão/controle do dispositivo. Representa as saídas de controle que são utilizadas para a mudança de estado em um processo, por exemplo, fechamento do disjuntor, abertura da chave, ajuste do *setpoint*, etc. Os atributos da classe são:

- **value**: indica o valor que representa a saída do atuador.
- **timeStamp**: indica a data e hora que último comando de controle foi efetivado.

- **operationInProgress**: indica, se verdadeiro, que o comando de controle que ainda não foi concluído (operação em progresso).
- **maxValue**: indica o valor máximo do **Control.value**.
- **minValue**: indica o valor mínimo de variação do **Control.value**.
- **dataType**: especifica o tipo de dado numérico do **Control.value**.
- **normalValue**: indica o valor normal da medição.

A.1.7.2. Pacote *Meas*: Classe – **ControlType**

Classe derivada de Naming (Pacote *Core*). Especifica o tipo de controle, por exemplo, o controle do disjuntor ligado/desligado (*BreakerOn/Off*), ajuste da referência da tensão do gerador (*GeneratorVoltageSetPoint*) etc. O atributo **ControlType.name** deve ser único entre todos os tipos especificados e descreve o tipo. O atributo **ControlType.name** é indicado para ser utilizado como localizador.

A.1.7.3. Pacote *Meas*: Classe – **Limit**

Subclasse de Naming (Pacote *Core*). Especifica um valor limite para uma medida. Uma medida normalmente tem diversos limites que são mantidos pela classe **LimitSet**. O real significado e uso de uma ocorrência (instância) de um limite (isto é, se o limite é de um alarme ou de um aviso, ou se é um limite máximo ou mínimo) não é capturado na classe Limit. Entretanto, o nome de uma ocorrência (instância) pode indicar ambos, seu significado e uso. O atributo da classe é:

- **value**: indica o valor a ser supervisionado.

A.1.7.4. Pacote *Meas*: Classe – **LimitSet**

Subclasse de Naming (Pacote *Core*). Especifica um conjunto de limites que estão associados com uma medida. Uma medida pode ter diversos **LimitSets** correspondendo a limites sazonais ou outras condições de mudança. A condição é capturada nos atributos de nome e descrição. O mesmo **LimitSet** pode ser usados por várias medidas. Em particular, limites de percentuais são utilizados desta maneira. O atributo da classe é:

- **isPercentageLimits**: é verdadeiro, se os valores limites são expressos em percentagem do valor normal.

A.1.7.5. Pacote *Meas*: Classe – **Measurement**

Subclasse de Naming (Pacote *Core*). Representa alguma medida ou quantidade. Qualquer equipamento pode ter medições associadas, por exemplo, uma subestação pode ter medições de temperatura e indicações de portas abertas, um transformador por ter medidas da temperatura do óleo e da pressão do tanque, e um disjuntor pode conter indicação do estado da chave. O relacionamento *PSR-Measurement* pretende capturar a utilização desta medição que é incluída na hierarquia de nomes baseada em EquipmentContainer. A hierarquia de nomes normalmente possui as medições das “folhas” (analogamente, o nível mais baixo de uma árvore binária), por exemplo, *Substation-VoltageLevel-Bay-Switch-Measurement*, esta medida representa o estado da chave. Algumas medidas representam quantidades relacionadas a uma particular posição de um sensor, por exemplo, um transformador de potencial (PT) no barramento ou um transformador de corrente (CT) no barramento entre um disjuntor e um alimentador o local de medição não é capturado na associação *PSR-Measurement*. Ao invés

disso, é capturado na associação *Measurement-Terminal* que torna possível definir o local onde o foi adquirido. O lugar é definido pela conexão do Terminal ao ConductingEquipment.

Existem duas ligações possíveis:

1. Measurement-Terminal-ConnectivityNode-Terminal-ConductingEquipment;
2. Measurement-Terminal-ConductingEquipment

A alternativa 2, é o único uso permitido. O uso da *Measurement-PSR* e *Measurement-Terminal* são permitidos ao mesmo tempo para uma medida particular. *Measurement-PSR* define as medições na hierarquia de nomes e *Measurement-Terminal* define onde a medida é colocada na topologia da rede. Os atributos da classe são:

- **positiveFlowIn**: indica, se verdadeiro, que a medida é de MW, MVAR ou AMPS com a convenção de valor positivo medido no Terminal, ou seja potência fluindo para dentro do PowerSystemResource relacionado.
- **maxValue**: valor máximo medido para algum **MeasurementValue.values**. Usado para a escala em gráficos de barra ou valores brutos de telemetria.
- **minValue**: valor mínimo medido para algum **MeasurementValue.values**. Usado para a escala em gráficos de barra ou valores brutos de telemetria.
- **dataType**: indica um dos diversos tipos numéricos (*INT_TYPE*, *UNSIGNED_TYPE*, *COMPLEX_TYPE* ...), que especifica o tipo de dados para as medições nominal, mínimo, máximo, etc., valor de MeasurementValue.
- **normalValue**: armazena o valor normal da medida.

A.1.7.6. Pacote *Meas*: Classe – **MeasurementType**

Subclasse de Naming (Pacote *Core*). Identifica a natureza da medida, por exemplo, temperatura interna (*IndoorTemperature*), temperatura externa (*OutDoorTemperature*), tensão do barramento (*BusVoltage*), tensão do gerador (*GeneratorVoltage*), fluxo da linha (*LineFlow*), etc. O atributo **MeasurementType.name** deve ser único entre todos os tipos especificados e descreve o tipo. O atributo **MeasurementType.aliasName** deve ser utilizado como localizador.

A.1.7.7. Pacote *Meas*: Classe – **MeasurementValue**

Subclasse de Naming (Pacote *Core*). Representa o estado atual de uma medição, ou seja uma ocorrência (instância) de uma medida de uma fonte específica. Medições podem ser associadas com diversos valores de estado, cada um representando uma diferente fonte de medida. Os atributos de classe são:

- **value**: indica o valor de uma medição.
- **timeStamp**: indica a data e hora de quando o último valor foi atualizado.
- **sensorAccuracy**: indica erro limite do sensor quando utilizado nas condições de referência. O limite é expresso como uma percentagem do valor máximo do sensor.

A.1.7.8. Pacote *Meas*: Classe – **MeasurementValueQuality**

Subclasse de Quality61850. São indicadores de qualidade das medições. Os Bits⁹ de 0 a 10 são definidos para a automação da subestação no **draft IEC 61850 part 7-3** [15]. Os bits de 11 a 15 são reservados para futuras implementações do referido documento. Os Bits de 16 a 31 são reservados para aplicações EMS (*Energy Management Systems*).

A.1.7.9. Pacote *Meas*: Classe – **MeasurementValueSource**

Classe derivada de Naming (Pacote *Core*). Descreve as fontes de atualização alternativa de um MeasurementValue.

A.1.7.10. Pacote *Meas*: Classe – **Quality61850**

A classe representa os indicadores de qualidade que estão definidas na especificação **IEC 61850** [15], exceto pela substituição por valor estimado (*estimatorReplaced*), que foi incluído nesta classe por questão de conveniência. Os atributos da classe são:

- **validity**: indica o estado (status) de validade da medição, que pode ser: boa, questionável ou inválida {*GOOD*, *QUESTIONABLE*, *INVALID*}.
- **overflow**: é verdadeiro, se o valor da medição excede a capacidade de ser devidamente representada. Por exemplo, o estouro do valor máximo de um contador, faz com que ele retorne ao valor zero.
- **outOfRange**: é verdadeiro, se o valor da medição exceder o valor predefinido da escala.

⁹ Bit, dígito binário (*binary digit*) – é a menor unidade de dados num computador, pode conter o valor 0 ou 1.

- **badReference**: é verdadeiro, se o valor da medição estiver incorreto por uma referência fora de calibragem.
- **oscillatory**: é verdadeiro, se houver oscilações. Para impedir alguma sobrecarga de comunicação, é sensato detectar e suprimir as entradas binárias (*binary inputs*) das oscilações. Se o sinal mudar em um determinado tempo (*tosc – time of oscillation*), duas vezes na mesma direção (de 0 para 1 ou de 1 para 0), então a oscilação é detectada e o identificador de qualidade (*oscillatory*) é ajustado. Se for detectado um número superior ao configurado para mudanças transientes, nesse momento o estado (*status*) de validade é ajustado para “questionável”. Se após o número de mudanças definido, o sinal ainda estiver oscilando, o valor será ajustado para estado oposto do valor estável precedente ou para um valor predefinido (*default*). Nesse caso o estado da validade “questionável” é reiniciado, e “inválido” é ajustado enquanto o sinal estiver oscilando. Se é configurado de tal modo, que nenhuma mudança transiente, seja detectada, então o estado de validade é ajustado imediatamente, para “inválido” ao mesmo tempo que o identificador de qualidade é marcado “oscilatório” (usado somente para informação de estado).
- **failure**: é verdadeiro, se a função de supervisão detectar uma falha interna ou externa. Por exemplo, falha de comunicação.
- **oldData**: é verdadeiro, se o valor da medida é antigo e possivelmente inválido, indicando que sua atualização não foi bem sucedida no intervalo de tempo especificado.
- **suspect**: é verdadeiro, se a função de correlação detectou que o valor medido não está consistente com os outros valores. Normalmente é um valor calculado por um estimador de estado da rede.

- **test**: é verdadeiro, se o valor da medição foi transmitido com a finalidade de testes.
- **operatorBlocked**: é verdadeiro, se o valor da medição está bloqueado e consequentemente indisponível para transmissão.
- **source**: indica a fonte relacionada a origem do valor medido. O valor pode ser adquirido do processo padrão ou adotado o valor de referência por outro substituto {*PROCESS*, *DEFAULTED*, *SUBSTITUTED*}.
- **estimatorReplaced**: é verdadeiro, se o valor foi substituído pelo estimador de estado. O **estimatorReplaced** não é um bit de qualidade na **IEC61850-7-3** [15], mas foi incluído nesta classe por questão de conveniência.

A.1.7.11. Pacote *Meas*: Classe – ValueAliasSet

Subclasse de Naming (Pacote *Core*). Descreve a tradução de um conjunto de valores em um nome. Cada **ValueAliasSet** tem um nome, uma descrição, etc. Uma medida específica pode representar um estado discreto como Aberto, Fechado, Intermediário, etc. Isto requer uma tradução do número do **MeasurementValue.value** para uma cadeia de caracteres, por exemplo, 0 = “inválido”, 1 = “aberto”, 2 = “fechado”, 3 = “intermediário”.

A.1.7.12. Pacote *Meas*: Classe – ValueToAlias

Subclasse de Naming (Pacote *Core*). Descreve a tradução de um valor particular em um nome, por exemplo, 1 = “aberto”.

A.1.8. Pacote *Outage*: Diagrama de Classes

A Figura 35, apresenta o Diagrama de Classes do pacote *Outage*.

Figura 35. Diagrama de Classes do pacote *Outage* - [6]

A.1.8.1. Pacote *Outage*: Classe – *ClearanceTag*

Classe derivada de Naming (Pacote *Core*). Representa uma etiqueta de liberação que é utilizada para autorizar e programar trabalho em um equipamento energizado em campo. O equipamento etiquetado estará indisponível para serviço comercial. Os atributos da classe são:

- **authorityName**: indica o nome da pessoa que esta autorizada a emitir a etiqueta.

- **clearanceTagType**: indica o tipo de etiqueta, dependendo do propósito de trabalho a ser executado e/ou tipo de controle de supervisão permitido.
- **deenergizeReqFlag**: verdadeiro, se o equipamento deve estar sem energia.
- **groundReqFlag**: verdadeiro, se o equipamento deve estar aterrado.
- **phaseCheckReqFlag**: verdadeiro, se a fase do equipamento deve ser verificada.
- **tagIssueTime**: indica data e hora de emissão da etiqueta.
- **workDescription**: descrição do trabalho a ser executado.
- **workEndTime**: indica o data e hora que a etiqueta de liberação está programada ser removida.
- **workStartTime**: indica o data e hora de início dos trabalhos.

A.1.8.2. Pacote *Outage*: Classe – *OutageSchedule*

Classe derivada de CurveSchedule (Pacote *Core*). Representa o período de tempo que um equipamento estará fora de serviço, por exemplo, para manutenção ou testes. Inclui o MW do equipamento durante a manutenção. O eixo X representa o tempo absoluto e o eixo Y representa o MW disponível no equipamento enquanto estiver fora de serviço.

A.1.8.3. Pacote *Outage*: Classe – *SwitchingOperation*

Classe derivada de Naming (Pacote *Core*). É utilizada para definir manobras em cada chave, necessárias para colocar o equipamento fora de serviço. Este processo pode estar associado com um outro item da subestação tal como um transformador, linha, ou gerador, ou chave

interruptora. Uma chave poder ser referenciada por diversas programações de desligamento.

Os atributos da classe são:

- **operationTime**: indica o data e hora de operação.
- **newState**: indica o novo estado da chave: aberta ou fechada *{open, close}*.

A.1.9. Pacote *Protection*: Diagrama de Classes

A Figura 36, mostra o Diagrama de Classes do Pacote *Protection*.

Figura 36. Diagrama de Classes do Pacote *Protection* - [6]

A.1.9.1. Pacote *Protection*: Classe – **CurrentRelay**

Subclasse de **ProtectionEquipment**. Representa um dispositivo que verifica o valor do fluxo da corrente em qualquer direção ou em determinado sentido. Os atributos da classe são:

- **currentLimit1**: indica o limite 1 (primeiro) da corrente, para a curva de tempo inverso.
- **currentLimit2**: indica o limite 2 (segundo) da corrente, para a curva de tempo inverso.
- **CurrentLimit3**: indica o limite 3 (terceiro) da corrente, para a curva de tempo inverso.

- **inverseTimeFlag**: é verdadeiro, se a corrente do relé possui característica de tempo inverso.
- **timeDelay1**: indica o atraso 1 (primeiro) de tempo inverso, para o limite 1 (primeiro) da corrente.
- **timeDelay2**: indica o atraso 2 (segundo) de tempo inverso, para o limite 2 (segundo) da corrente.
- **TimeDelay3**: indica o atraso 3 (terceiro) de tempo inverso, para o limite 3 (terceiro) da corrente.

A.1.9.2. Pacote *Protection*: Classe – ProtectionEquipment

Subclasse de Equipment. Representa um dispositivo elétrico para responder a determinadas condições de entrada de maneira prescrita, e depois que as condições especificadas são atingidas, para causar a operação de contato ou mudança abrupta similar em circuitos de controle elétrico associado, ou simplesmente exibir a condição detectada. Equipamentos de proteção estão associados a equipamentos condutores, e normalmente operam disjuntores de circuitos. Para os relés de proteção, o atributo **typeName** é um tipo de identificador, conforme especificado no padrão IEC 61850, com um prefixo “PRO_”. Por exemplo, o relé de sobretensão (**OverVoltage Relay**) é identificado como: “PRO_OVR”. Os atributos da classe são:

- **relayDelayTime**: indica o tempo de atraso de detecção (em segundos), da operação do relé em condições anormais.
- **highLimit**: indica o valor máximo permitido.

- **lowLimit**: indica o valor mínimo permitido.
- **powerDirectionFlag**: é verdadeiro, se o sentido é o mesmo que o valor do fluxo de potência ativa.

A.1.9.3. Pacote *Protection*: Classe – RecloseSequence

Subclasse de Naming (Pacote *Core*). Representa uma sequência de religamento (*reclose sequence*) que é definida para cada possibilidade de religamento de um disjuntor. Os atributos da classe são:

- **recloseDelay**: indica o lapso de tempo antes do início da etapa do religamento.
- **recloseStep**: indica a posição ordinal da etapa do religamento relativa às outras etapas da sequência.

A.1.9.4. Pacote *Protection*: Classe – SynchrocheckRelay

Subclasse de ProtectionEquipment. Representa um dispositivo que opera quando dois circuitos de corrente alternada estiverem dentro de limites desejáveis da frequência, ângulo de fase e tensão, para permitir ou causar o paralelismo destes dois circuitos. Utilizado para impedir o paralelismo de ilhas topológicas não síncronas. Os atributos da classe são:

- **maxAngleDiff**: indica a diferença máxima admissível do ângulo de fase do vetor de tensão, quando o dispositivo está aberto.
- **maxFreqDiff**: indica a diferença máxima admissível da frequência, quando o dispositivo está aberto.

- **maxVoltDiff**: indica a diferença máxima admissível de tensão, quando o dispositivo está aberto.

A.1.10. Pacote *Topology*: Diagrama de Classes

A Figura 37, apresenta o Diagrama de Classes do Pacote *Topology*.

Figura 37. Diagrama de Classes do Pacote *Topology* - [6]

A.1.10.1. Pacote *Topology*: Classe – *ConnectivityNode*

Classe derivada de Naming (Pacote *Core*). Representa os nós de conectividade, que são os pontos com impedância zero, onde os terminais do equipamento condutor são juntamente conectados.

A.1.10.2. Pacote *Topology*: Classe – *TopologicalIsland*

Classe derivada de Naming (Pacote *Core*). Representa um subconjunto da rede eletricamente conectado. As ilhas topológicas podem mudar conforme alteração do estado atual da rede (por exemplo, chaves desligadas, disjuntores, etc. modificam o estado da rede).

A.1.10.3. Pacote *Topology*: Classe – *TopologicalNode*

Classe derivada de Naming (Pacote *Core*). Representa um conjunto de nós de conectividade, no estado atual da rede, que estão ligados entre si através de chaves fechadas. Nós topológicos podem mudar conforme alteração do estado atual da rede (por exemplo, chaves, disjuntores, etc. mudam o estado da rede). Os atributos da classe são:

- **energized**: é verdadeiro, se o nó está energizado.
- **loadCarrying**: é verdadeiro, se o nó está conduzindo corrente de carga.
- **netInjectionMVar**: valor da injeção líquida de MVar.
- **netInjectionMW**: valor da injeção líquida de MW.
- **observabilityFlag**: é verdadeiro, se o estado do nó é “observável”.
- **phaseAngle**: indica o ângulo de fase do nó.

- **voltage**: indica a tensão do nó.

A.1.11. Pacote *Wires*: Diagrama de Classes

O pacote *Wires* é composto por seis diagramas de classes, a saber: *InheritanceHierarchy*, *LineModel*, *EquipmentContainment*, *TransformerModel*, *RegulatingEquipment*. e *VoltageControl*.

O Diagrama de Classes do Pacote *Wires* / *InheritanceHierarchy*, é apresentado na Figura 38.

Figura 38. Diagrama de Classes do Pacote *Wires* / *InheritanceHierarchy* - [6]

A Figura 39, apresenta o Diagrama de Classes do Pacote *Wires / LineModel*.

Figura 39. Diagrama de Classes do Pacote *Wires / LineModel* - [6]

O Diagrama de Classes do Pacote *Wires / EquipmentContainment*, é mostrado na Figura 40.

Figura 40. Diagrama de Classes do Pacote *Wires / EquipmentContainment* - [6]

A Figura 41, apresenta o Diagrama de Classes do Pacote *Wires / TransformerModel*.

Figura 41. Diagrama de Classes do Pacote *Wires / TransformerModel* - [6]

A Figura 42, apresenta o Diagrama de Classes do Pacote *Wires / RegulatingEquipment*.

Figura 42. Diagrama de Classes do Pacote *Wires / RegulatingEquipment* - [6]

A Figura 43, apresenta o Diagrama de Classes do Pacote *Wires / VoltageControl*.

Figura 43. Diagrama de Classes do Pacote *Wires / VoltageControl* - [6]

A.1.11.1. Pacote *Wires*: Classe – **ACLineSegment**

Subclasse de Conductor. Representa um fio ou uma combinação de fios, com características elétricas consistentes, construindo um único sistema elétrico, utilizado para transportar corrente alternada entre pontos de um sistema de potência. Os atributos são os mesmos especificados na superclasse.

A.1.11.2. Pacote *Wires*: Classe – **Breaker**

Subclasse de Switch. Representa um dispositivo mecânico de chaveamento capaz de suportar, e interromper correntes nas condições normais de circuito e também suportar por um tempo específico, e interromper correntes sob condições anormais específicas de um circuito, por exemplo, as de curto-circuito. O atributo **typeName** indica o princípio de funcionamento

dispositivo chaveador, a óleo, rajada de ar, vácuo e SF6 (hexafluoreto de enxofre). Os atributos da classe são:

- **ampRating**: indica o valor da corrente de interrupção em falta, em Ampères.
- **inTransitTime**: indica o tempo de transição da abertura ao fechamento, em segundos.

A.1.11.3. Pacote *Wires*: Classe – *BusBarSection*

Subclasse de Connector. Representa um condutor, ou grupo de condutores, com impedância desprezível, que serve para conectar outros equipamentos condutores dentro de uma subestação. Medidas de tensão são normalmente obtidas dos transformadores de tensão que estão conectados aos segmentos do barramento. Uma seção do barramento pode ter vários terminais físicos, porém para análise é modelado com exatamente um terminal lógico. O atributo **typeName** indica o tipo de segmento do barramento, por exemplo: principal (*main*), ou transferência (*transfer*). Os atributos da classe são os mesmos da superclasse.

A.1.11.4. Pacote *Wires*: Classe – *Compensator*

Subclasse de RegulatingCondEq. Representa um banco de capacitores chaveáveis, série ou *shunt*. A aplicação série ou paralela de um banco de capacitores é determinada por sua conectividade na rede. Os atributos da classe são:

- **aVRDelay**: indica o tempo de atraso em segundos, requerido pelo dispositivo ao ser ligado ou desligado pelo regulador automático de tensão (AVR – *Automatic Voltage Regulation*).
- **impedance**: indica a impedância de sequência positiva do capacitor.

- **maximumkV**: indica a tensão máxima na qual o banco de capacitores deve operar.
- **maximumSections**: indica o número máximo de seções de um banco de capacitores que podem ser operados.
- **minimumkV**: indica a tensão mínima na qual um banco de capacitores deve operar.
- **mVArPerSection**: indica a magnitude em MVar, de cada seção comutável de um banco de capacitores, na tensão nominal.
- **nominalkV**: indica a tensão nominal em que o MVar nominal foi medido. Deve estar normalmente dentro de 10% da tensão, da rede na qual o capacitor é ligado.
- **nominalMVar**: indica o MVar nominal do banco de capacitores na tensão nominal. Este número deve ser positivo.
- **normalSections**: indica o número de seções comutáveis dentro de um banco de capacitores. Este número deve corresponder ao MVar nominal.
- **r**: indica a resistência de seqüência positiva do banco de capacitores.
- **switchOnCount**: é um contador do número de vezes que o banco foi ligado, desde a última vez em que o contador do capacitor foi reajustado ou inicializado.
- **switchOnDate**: indica a data e hora de quando o banco de capacitores foi ligado pela última vez.
- **voltSensitivity**: indica a sensibilidade de tensão requerida na regulação da tensão de um barramento, em (kV por unidade/MVar).

- **x**: indica a reatância de seqüência positiva do banco de capacitores.
- **yPerSection**: indica a admitância de cada seção comutável de um banco de capacitores. É calculada utilizando o MVAr por seção, e corrigida pela tensão da rede.
- **compensatorType**: utilizado para indicar a quantidade de terminais. Um compensador de derivação tem somente um terminal (terra implícito), enquanto compensadores em série possuem dois terminais *{shunt, series}*.

A.1.11.5. Pacote *Wires*: Classe – **CompositeSwitch**

Classe derivada de EquipmentContainer (Pacote *Core*). Representa o modelo de um conjunto de interruptores normalmente fechados dentro de um mesmo gabinete, possivelmente com inter-travamentos que restringem a combinação de posições do interruptor. Estes são encontrados tipicamente em redes de distribuição de média tensão. Um CompositeSwitch poderia representar uma unidade principal em anel (RMU – *Ring-Main-Unit*), ou um mecanismo de chaveamento montado num bloco, com os dispositivos internos primitivos tais como, o barramento interno e mais 3 ou 4 interruptores internos, cada qual podendo, individualmente, ser aberto ou fechado. Um CompositeSwitch é um conjunto de interruptores nele contido, podem ser utilizados para representar uma chave multi-posicional, que pode estar aberta, conectada ao terra, ou ao barramento. Os atributos da classe são:

- **compositeSwitchType**: é um código alfanumérico que pode ser usado como referência a uma informação extra, tal como a descrição do esquema de inter-travamento, se houver algum.

A.1.11.6. Pacote *Wires*: Classe – Conductor

Classe derivada de Naming (Pacote *Core*). Representa a combinação de material condutor com características elétricas consistentes, construindo um único sistema elétrico, utilizado para transportar corrente entre pontos no sistema de potência. Os atributos da classe são:

- **b0ch**: indica a susceptância para terra (carga) de seqüência zero, distribuída uniformemente, em todo o trecho da linha.
- **bch**: indica a susceptância para terra (carga) de seqüência positiva, distribuída uniformemente, em todo o trecho da linha.
- **g0ch**: indica a condutância para terra (carga) de seqüência zero, distribuída uniformemente, em todo o trecho da linha.
- **gch**: indica a condutância para terra (carga) de seqüência positiva, distribuída uniformemente, em todo o trecho da linha.
- **length**: indica o comprimento do segmento, para cálculo da capacidade do trecho de linha.
- **r**: indica a resistência de seqüência positiva de todo o trecho da linha.
- **r0**: indica a resistência de seqüência zero de todo o trecho da linha.
- **x**: indica a reatância de seqüência positiva de todo o trecho da linha.
- **x0**: indica a reatância de seqüência zero de todo o trecho da linha.

A.1.11.7. Pacote *Wires*: Classe – **ConductorType**

Subclasse de Naming. Representa um fio ou cabo condutor (pela especificação do IEEE). Um tipo específico de cabo ou uma combinação de cabos não isolados entre si, apropriados para conduzir a corrente elétrica. Pode estar desencapado ou isolado. Os atributos da classe são:

- **sheathResistance**: indica a resistência da blindagem dos cabos condutores.
- **SheathReactance**: indica a reatância da blindagem dos cabos condutores.

A.1.11.8. Pacote *Wires*: Classe – **Connector**

Subclasse de ConductingEquipment (Pacote *Core*). Representa um condutor, ou grupo de condutores com impedância desprezível, que serve para conectar outros equipamentos condutores dentro de uma subestação que são modelados com um único terminal lógico. Os atributos da classe são os mesmos da superclasse.

A.1.11.9. Pacote *Wires*: Classe – **DCLineSegment**

Subclasse de Conductor. Representa um fio ou uma combinação de fios não isolados, com características elétricas consistentes, utilizados para transportar a corrente contínua entre pontos da região de corrente contínua do sistema de potência. Os atributos da classe são:

- **dcSegmentInductance**: Indutância do segmento da linha de corrente contínua, em mH (milihenry).
- **dcSegmentResistance**: Resistência do segmento da linha de corrente contínua, em ohms.

A.1.11.10. Pacote *Wires*: Classe – **Disconnecter**

Subclasse de Switch. Representa um dispositivo mecânico de chaveamento operado manualmente ou por motor, utilizado para mudar as conexões de um circuito, ou para isolar um circuito ou equipamento de sua fonte de energia. Os atributos são os mesmos da superclasse.

A.1.11.11. Pacote *Wires*: Classe – **EnergyConsumer**

Subclasse de ConductingEquipment (Pacote *Core*). Representa o usuário genérico de energia, um ponto de consumo no modelo do sistema de potência. Os atributos da classe são:

- **conformingLoadFlag**: o valor é verdadeiro, se a carga está adequada. Isto é, segue a carga da área a qual o consumidor pertence.
- **customerCount**: indica o número de clientes individuais representados por esta demanda.
- **pFexp**: indica o expoente da frequência (por unidade) da potência produzida.
- **pFixed**: indica a componente real da carga que é uma quantidade fixa em MW.
- **pFixedPct**: indica a percentagem de MW fixo do grupo de carga.
- **pnom**: indica o valor nominal da potência, em MW. A potência nominal real é ajustada de acordo com o perfil de carga selecionado para o consumidor. É igual a um, no perfil de carga.
- **pnomPct**: indica a percentagem de MW nominal do MW do grupo de carga.

- **powerFactor**: indica o fator de potência para a parcela nominal da carga. Definido como MW/MVA.
- **pVexp**: indica o expoente da tensão (por unidade) da potência real produzida.
- **qFexp**: indica o expoente da frequência (por unidade) da potência reativa produzida.
- **qfixed**: indica a componente reativa da carga que é uma quantidade fixa em MVar.
- **qfixedPct**: indica a percentagem de MVar do grupo fixo de carga.
- **qnom**: indica o valor nominal da potência reativa em MVar.
- **qnomPct**: indica a percentagem de MVar nominal do grupo de carga.
- **qVexp**: indica o expoente da tensão (por unidade) da potência reativa produzida.

A.1.11.12. Pacote *Wires*: Classe – *EquivalentSource*

Subclasse de ConductingEquipment (Pacote *Core*). Representa o equivalente genérico para um fornecedor de energia em determinado nível de tensão de transmissão ou distribuição. Os atributos da classe são:

- **xn**: indica a reatância de *Thevenin* de seqüência negativa em ohms.
- **rn**: indica a resistência de *Thevenin* de seqüência negativa em ohms.
- **nominalVoltage**: indica a tensão nominal fase-fase.
- **x**: indica a reatância de *Thevenin* de seqüência positiva.
- **r**: indica a resistência de *Thevenin* de seqüência positiva.

- **voltageAngle**: indica o ângulo de uma fase em circuito aberto em radianos.
- **voltageMagnitude**: indica a magnitude de tensão fase-fase em circuito aberto.
- **x0**: indica a reatância de *Thevenin* de seqüência zero em ohms.
- **r0**: indica a resistência de *Thevenin* de seqüência zero em ohms.

A.1.11.13. Pacote *Wires*: Classe – Fuse

Subclasse de Switch. Representa um dispositivo protetor de sobrecorrente com um elemento fundível que torna o circuito aberto quando fortemente aquecido pela passagem da corrente através dele. Um fusível é considerado um dispositivo de chaveamento porque ele interrompe a corrente, tal como um disjuntor. O atributo da classe é:

- **ampRating**: indica o valor da corrente de interrupção, em Ampères.

A.1.11.14. Pacote *Wires*: Classe – Ground

Subclasse de ConductingEquipment (Pacote *Core*). Representa o ponto comum para conectar à terra equipamentos condutores, tais como, capacitores de derivação (desvio). O atributo **typeName** indica o tipo de terra: malha ou haste terra. Recomenda-se usar a classe GroundDisconnector em vez de Ground quando aplicar o CIM. No caso de aterramento de um compensador de derivação, utilizar o compensador do tipo *SHUNT*. Os atributos da classe são os mesmos da superclasse.

A.1.11.15. Pacote *Wires*: Classe – GroundDisconnector

Subclasse de Switch. Representa um dispositivo de chaveamento mecânico operado manualmente ou por motor, utilizado para isolar um circuito ou equipamento do terra. Os atributos são os mesmos da superclasse.

A.1.11.16. Pacote *Wires*: Classe – HeatExchanger

É uma subclasse de Equipment (Pacote *Core*). Representa o equipamento de refrigeração e extração de calor do equipamento elétrico. Os atributos são os mesmos especificados na superclasse.

A.1.11.17. Pacote *Wires*: Classe – Jumper

Subclasse de Switch. Representa um trecho curto de condutor com impedância desprezível, o qual pode ser manualmente removido e substituído se o circuito estiver sem energia. Note que ramos de impedância zero podem ser modelados por um ACLLineSegment com ConductorType de impedância zero. Os atributos são os mesmos especificados na superclasse.

A.1.11.18. Pacote *Wires*: Classe – Junction

Subclasse de Connector. Representa um ponto onde um ou mais equipamentos condutores estão conectados com resistência zero (nula). Os atributos da classe são os mesmos da superclasse.

A.1.11.19. Pacote *Wires*: Classe – Line

Subclasse derivada de PowerSystemResource (Pacote *Core*). Representa uma parte do componente de um sistema que se estende entre subestações vizinhas ou de uma subestação a um ponto adjacente de conexão. Os atributos são os mesmos especificados na superclasse.

A.1.11.20. Pacote *Wires*: Classe – LoadBreakSwitch

Subclasse de Switch. Representa um dispositivo mecânico capaz de suportar e interromper correntes em condições normais de operação. O atributo da classe é:

- **ampRating**: indica a capacidade de transporte de corrente, expressa em Ampères, de um fio ou cabo, sob condições térmicas determinadas.

A.1.11.21. Pacote *Wires*: Classe – MVarCapabilityCurve

Subclasse de CurveSchedule. Representa a envoltória da curva de potência reativa contra a potência ativa de uma máquina síncrona, nas modalidades de geração e motorização. Para cada valor de MW, existe um valor limite máximo e mínimo de MVar correspondente. Normalmente, haverá uma curva separada para cada condição do líquido refrigerante, tal como a pressão de hidrogênio. Os atributos da classe são:

- **coolantTemperature**: indica a temperatura do líquido refrigerante da máquina, em graus Celsius (por exemplo: ar ambiente ou a água de circulação do estator).
- **hydrogenPressure**: indica a pressão do hidrogênio de refrigeração.

A.1.11.22. Pacote *Wires*: Classe – **PowerTransformer**

É uma classe derivada de Equipment (Pacote *Core*). Representa um dispositivo elétrico que consiste de dois ou mais enrolamentos acoplados, com ou sem núcleo magnético, para permitir ligações entre circuitos elétricos. Transformadores podem ser utilizados para controlar a tensão e a mudança de fase (fluxo de MW). Os atributos da classe são:

- **bmagSat**: indica, em percentagem, a susceptância de magnetização do núcleo na região de saturação.
- **magBaseKV**: indica a tensão de base na qual as medidas de saturação de magnetização foram feitas.
- **magSatFlux**: indica, em percentagem, o fluxo de saturação de magnetização do núcleo.
- **phases**: descreve as fases em que o transformador de potência está conectado. Os possíveis valores são: ABCN, ABC, ABN, ACN, BCN, AB, AC, BC, AN, BN, CN, A, B, C, N.
- **transfCoolingType**: indica o tipo de refrigeração do transformador.
- **transformerType**: indica o tipo do transformador *{fix, voltageControl, phaseControl, voltageAndPhaseControl}*.

A.1.11.23. Pacote *Wires*: Classe – **RectifierInverter**

Subclasse de ConductingEquipment (Pacote *Core*). Representa o equipamento de conversão bidirecional de corrente contínua/corrente alternada. Pode ser utilizado para controlar a

corrente contínua (DC – *Direct Current*), a tensão da corrente contínua, o fluxo de potência da corrente contínua, ou ângulo de disparo. Os atributos da classe são:

- **ratedKV**: indica a tensão de base primária do retificador/inversor.
- **bridges**: indica o número de pontes.
- **commutatingReactance**: indica a reatância de inversão em ohms, na frequência do barramento de corrente alternada.
- **commutatingResistance**: indica a resistência de inversão em ohms.
- **compoundResistance**: indica a resistência composta em ohms.
- **minCompoundVoltage**: indica o valor mínimo da componente DC.

A.1.11.24. Pacote *Wires*: Classe – *RegulatingCondEq*

Subclasse de ConductingEquipment (Pacote *Core*). Representa um tipo de equipamento condutor que pode regular medições e ter uma RegulationSchedule (programação de regulação). Os atributos da classe são os mesmos da superclasse.

A.1.11.25. Pacote *Wires*: Classe – *RegulationSchedule*

Classe derivada de CurveSchedule (Pacote *Core*). Representa um padrão preestabelecido de tempo excedente para uma variável controlada, por exemplo, tensão do barramento. Os atributos da classe são:

- **lineDropCompensation**: indica que a compensação de queda de tensão, deve ser aplicada.

- **lineDropR**: indica a resistência de ajuste.
- **LineDropX**: indica a reatância de ajuste.

A.1.11.26. Pacote *Wires*: Classe – **StaticVarCompensator**

Subclasse de RegulatingCondEq. Representa uma facilidade para suportar a variação e controle da potência reativa de derivação. O SVC (*StaticVarCompensator*) consiste normalmente de um transformador *stepdown*, um filtro, um reator controlado por *thyristor* e braços de capacitor chaveado por *thyristor*. Os atributos da classe são:

- **capacitiveRating**: indica a capacidade máxima disponível de potência reativa capacitiva.
- **inductiveRating**: indica a capacidade máxima disponível de potência reativa indutiva.
- **sVCControlMode**: indica o modo de controle do SVC – *StaticVarCompensator*: MVar ou tensão.
- **slope**: indica a razão da variação de tensão pela potência reativa (kV/MVar) por unidade.
- **voltageSetPoint**: indica o valor de referência (*setpoint*) da tensão (kV).

A.1.11.27. Pacote *Wires*: Classe – **Switch**

É uma classe derivada de ConductingEquipment (Pacote *Core*). Representa um dispositivo genérico projetado para fechar, ou abrir, ou ambos, um ou mais circuitos elétricos. Os atributos da classe são:

- **normalOpen**: indica, se verdadeiro, que o dispositivo de chaveamento está normalmente aberto.

- **switchOnCount**: é um contador de chave ligada, desde a última vez em que o interruptor foi reajustado ou inicializado.
- **switchOnDate**: indica a data e hora de quando o interruptor foi ligado pela última vez.

A.1.11.28. Pacote *Wires*: Classe – **SynchronousMachine**

Subclasse de RegulatingCondEq. Representa um dispositivo eletromecânico que opera de maneira sincronizada com a rede. Pode ser uma simples máquina operacional, um gerador, um condensador síncrono ou uma bomba. Os atributos da classe são:

- **aVRToManualLag**: indica o tempo de atraso, em segundos, que ocorre na mudança do AVR (*Automatic Voltage Regulation*) para o manual, desencadeando uma violação de atraso de MVar.
- **aVRToManualLead**: indica o tempo de atraso, em segundos, que ocorre na mudança do AVR para o manual.
- **baseMVar**: indica o valor inicial de MVar. Este valor representa o MVar inicial que pode ser usado por qualquer função aplicativa.
- **coolantCondition**: indica a temperatura ou pressão do meio de refrigeração.
- **coolantType**: indica o método de refrigeração da máquina, por exemplo, ar, gás hidrogênio ou água.
- **damping**: indica o coeficiente de torque de amortecimento, uma constante de proporcionalidade que, quando multiplicada pela velocidade angular de um rotor de pólos com respeito ao campo magnético (frequência), resulta em um torque de amortecimento.

- **inertia**: indica a energia armazenada no rotor quando operando na velocidade nominal. Este valor é usado no quadro de referência de potência de aceleração, para soluções no simulador de treinamento do operador.
- **manualToAVR**: indica o tempo de atraso, em segundos, que ocorre na mudança do manual para o AVR (*Automatic Voltage Regulation*). Este valor é utilizado no quadro de referência de potência de aceleração nos cálculos de fluxo de potência.
- **maximumKV**: indica o limite máximo de kV da unidade.
- **maximumMVar**: indica o limite máximo de MVar. Este é o limite máximo, de placa, da unidade.
- **minimumKV**: indica o limite mínimo de kV da unidade.
- **minimumMVar**: indica o limite mínimo de MVar da unidade.
- **r**: indica a resistência de seqüência positiva da máquina síncrona.
- **r0**: indica a resistência de seqüência zero da máquina síncrona.
- **ratedMVA**: indica o MVA nominal, de placa, da unidade.
- **x**: indica a reatância de seqüência positiva da máquina síncrona.
- **x0**: indica a reatância de seqüência zero da máquina síncrona.
- **xDirectSubtrans**: indica a reatância subtransitória de eixo direto, também conhecido como $X''d$.

- **xDirectSync**: indica a reatância síncrona do eixo direto. O quociente de um valor sustentado da componente de corrente alternada (AC) da tensão de armadura, que é produzida pelo fluxo total de eixo direto, devido à corrente de eixo direto da armadura e o valor da componente AC desta corrente, para a máquina funcionando na velocidade nominal (X_d).
- **xDirectTrans**: indica a reatância subtransitória de eixo direto, também conhecida como X'_d .
- **xQuadSubtrans**: indica a reatância subtransitória de eixo quadratura, também conhecida como X''_q .
- **xQuadSync**: indica a reatância síncrona de eixo quadratura (X_q), que é a razão entre a componente reativa da tensão de armadura, devido à componente de eixo quadratura da corrente de armadura, e este componente de corrente, nas condições de regime constante e na frequência nominal.
- **xQuadTrans**: indica a reatância transitória de eixo quadratura, também conhecida como X'_q .
- **operatingMode**: indica a modalidade de operação atual, isto é, gerador ou compensador $\{generator, condenser\}$.
- **type**: indica as modalidades de operação da máquina síncrona, isto é, como um gerador, um compensador, ou ambos $\{generator, condenser, generator_or_condenser\}$.
- **condenserMW**: indica a potência ativa do compensador em MW.

A.1.11.29. Pacote *Wires*: Classe – TapChanger

Classe derivada de PowerSystemResource (Pacote *Core*). Representa o mecanismo de mudança da posição do “tap” do enrolamento do transformador. Os atributos da classe são:

- **highStep**: indica o degrau mais elevado possível de posicionamento do “tap” do transformador.
- **initialDelay**: indica, para um LTC¹⁰ – *Load Tap Changing Transformer*, o atraso inicial da operação de mudança de “tap” (primeiro degrau de mudança).
- **lowStep**: indica o degrau mais baixo possível de posicionamento do “tap” do transformador.
- **neutralKV**: indica a tensão na posição neutra do “tap”.
- **neutralStep**: indica a posição do “tap” para o neutro.
- **normalStep**: indica o “tap” utilizado na operação normal da rede.
- **stepPhaseShiftIncrement**: indica a mudança de fase em graus. Um valor positivo, indica a mudança de fase positiva de posicionamento do “tap”, em relação ao outro enrolamento (para um transformador de dois enrolamentos), ou ao ponto “T” (para um transformador de três enrolamentos).
- **stepVoltageIncrement**: indica o incremento de tensão em cada “tap”, em percentagem da tensão nominal.

¹⁰ LTC (*Load Tap Changing*), os transformadores com LTC apresentam mecanismos automáticos para compensar a mudança de tensão sob condições de variação de carga.

- **subsequentDelay**: indica, para um LTC, o atraso (em segundos) da operação subsequente de mudança de “tap”.
- **tcuControlMode**: indica, para um LTC, a modalidade de controle de mudança de “tap”, por exemplo: *Off, Local, Volt, MVar*.

A.1.11.30. Pacote *Wires*: Classe – **TransformerWinding**

Classe derivada de ConductingEquipment. Representa o enrolamento que está associado com cada terminal de um transformador. Os atributos da classe são:

- **b**: indica a susceptância de magnetização do enrolamento secundário.
- **insulationKV**: nível de tensão de isolamento.
- **connectionType**: indica o tipo de conexão do enrolamento, por exemplo: *delta, estrela (em Y) e zigzag*.
- **emergencyMVA**: indica o MVA que o enrolamento pode carregar sob condições de emergência.
- **g**: indica a condutância de magnetização do enrolamento secundário.
- **grounded**: é verdadeiro, se o enrolamento for aterrado.
- **r**: indica a resistência de seqüência positiva do enrolamento.
- **r0**: indica a resistência de seqüência zero do enrolamento.
- **ratedKV**: indica a tensão nominal (fase-terra) do enrolamento.

- **ratedMVA**: indica o MVA nominal para o enrolamento.
- **rground**: indica a resistência de terra, conectado no ponto de aterramento do transformador.
- **shortTermMVA**: indica o MVA que o enrolamento pode carregar por um curto período de tempo.
- **windingType**: indica o tipo do enrolamento, por exemplo: primário, secundário, terciário ou quaternário.
- **x**: indica a reatância de sequência positiva do enrolamento.
- **x0**: indica a reatância de sequência zero do enrolamento.
- **xground**: indica a reatância de terra, conectado no ponto de aterramento do transformador.

A.1.11.31. Pacote *Wires*: Classe – VoltageControlZone

Subclasse derivada de PowerSystemResource (Pacote *Core*). Representa uma área da linha de transmissão ou sub-transmissão que é definida com o propósito de controle da tensão. Uma zona de controle de tensão consiste em um conjunto de subestações com os respectivos barramentos, cuja tensão será controlada. Os atributos são os mesmos especificados na superclasse.

A.1.11.32. Pacote *Wires*: Classe – WindingTest

Classe derivada de Naming (Pacote *Core*). São os dados de testes físicos do enrolamento de um transformador (ou desfasador), dos pares de enrolamento/*tap*. Estes dados de testes podem

ser utilizados para derivar outros atributos do transformador específico ou modelos desfasadores. Os atributos da classe são:

- **excitingCurrent**: indica o percentual de corrente de excitação (“para” o enrolamento de circuito aberto), do relatório de teste.
- **fromTapStep**: indica o número de degraus de *tap* do par de teste, “proveniente” do enrolamento.
- **leakageImpedance**: indica a impedância de dispersão medida entre o enrolamento em curto-circuito e todos os outros enrolamentos em circuito aberto. A impedância de dispersão é expressa em unidades baseadas nos valores de MVA e kV nominais do enrolamento.
- **loadLoss**: indica o kW de perda (“para” o enrolamento em curto-circuito), do relatório de teste.
- **noLoadLoss**: indica o kW de carga útil (“para” o enrolamento em circuito aberto), do relatório de teste.
- **phaseShift**: indica a desfasagem do ângulo de fase medida no circuito aberto “para” o enrolamento, com o conjunto do enrolamento “proveniente” para a tensão nominal do enrolamento “proveniente”, e todos outros enrolamentos em circuito aberto.
- **toTapStep**: indica o número de degraus de *tap* do par de teste, “para” o enrolamento.
- **voltage**: indica a tensão medida em circuito aberto “para” o enrolamento, com o conjunto de enrolamento “proveniente” para a tensão nominal do enrolamento “proveniente”, e

todos outros enrolamentos em circuito aberto.

A.1.11.33. Pacote *Wires*: Classe – WireArrangement

Subclasse de Naming. Representa a identificação, espaçamento e arranjo dos cabos de um ConductorType. Os atributos da classe são:

- **mountingPointX**: indica o ponto (coordenada X) de montagem, do primeiro cabo (cabo 1).
- **MountingPointY**: indica o ponto (coordenada Y) de montagem, do primeiro cabo (cabo 1).

A.1.11.34. Pacote *Wires*: Classe – WireType

Subclasse de Naming. Representa o cabo condutor (pela especificação do IEEE). Um tipo específico de cabo ou combinações de cabos, não isolados entre si, apropriado para conduzir corrente elétrica. Os atributos da classe são:

- **phaseConductorCount**: indica o número de cabos condutores não isolados que compõem um feixe simétrico (1-12).
- **phaseConductorSpacing**: indica o espaçamento entre os subcondutores não isolados dentro de um feixe simétrico.
- **ampRating**: indica a capacidade de transportar (suportar) a corrente, expressa em Ampères, do cabo ou subcondutor, sob condições térmicas determinadas.

- **gMR**: indica o Raio da Forma Geométrica ou Raio do Meio Geométrico (GMR – *Geometric Mean Radius*). Se o condutor for substituído por um tubo fino (conduíte) de raio GMR, então sua reatância será idêntica a reatância do condutor atual.
- **radius**: indica o raio do condutor.
- **resistance**: indica a resistência por unidade de comprimento do condutor.

A.2. IEC 61970-302

A.2.1. Pacote *EnergyScheduling*: Diagrama de Classes

A Figura 44, apresenta o Diagrama de Classes do pacote *EnergyScheduling*.

Figura 44. Diagrama de Classes do pacote *EnergyScheduling* - [6]

A.2.2. Pacote *Financial*: Diagrama de Classes

A Figura 45, apresenta o Diagrama de Classes do pacote *Financial*.

Figura 45. Diagrama de Classes do pacote *Financial* - [6]

A.2.3. Pacote *Reservation*: Diagrama de Classes

A Figura 46, apresenta o Diagrama de Classes do pacote *Reservation*.

Figura 46. Diagrama de Classes do pacote *Reservation* - [6]

A.3. IEC 61970-303

A.3.1. Pacote SCADA: Diagrama de Classes

A Figura 47, apresenta o Diagrama de Classes do pacote SCADA.

Figura 47. Diagrama de Classes do pacote SCADA - [6]

A.3.1.1. Pacote *SCADA*: Classe – CommunicationLink

Representa a conexão às unidades remotas através de um ou mais canais de comunicação. Canais de comunicação de dados redundantes podem existir. A classe CommunicationLink é derivada (herda características) de PowerSystemResource (Pacote *Core*). O propósito da classe é permitir medições dos canais de comunicação. Essas medições podem ser utilizadas para modelar o estado (*status*) do canal como operacional, fora de serviço, falha da unidade etc.

A.3.1.2. Pacote *SCADA*: Classe – RemoteControl

Classe derivada de RemotePoint. Representa as atuações das ações de controle remoto enviadas pela unidade remota aos atuadores do processo. Os atributos da classe são:

- **remoteControlled**: verdadeiro, se a unidade remota é controlada à distância.
- **actuatorMaximum**: indica o valor máximo do sinal (comando) do atuador.
- **actuatorMinimum**: indica o valor mínimo do sinal (comando) do atuador.

A.3.1.3. Pacote *SCADA*: Classe – RemotePoint

Classe derivada de Naming (Pacote *Core*). Corresponde aos valores medidos ou controlados remotamente, podendo ainda incluir valores calculados.

A.3.1.4. Pacote *SCADA*: Classe – RemoteSource

Classe derivada de RemotePoint. Representa as variáveis de estado que são processadas por telemetria, ou calculadas dentro de uma unidade remota. Os atributos da classe são:

- **sensorMaximum**: indica o valor máximo do item que a telemetria pode retornar.
- **sensorMinimum**: indica o valor mínimo do item que a telemetria pode retornar.
- **scanInterval**: indica o intervalo de tempo entre as varreduras.
- **deadband**: indica a menor variação no valor a ser informado.

A.3.1.5. Pacote *SCADA*: Classe – **RemoteUnit**

Representa uma unidade remota que pode ser uma RTU (*Remote Terminal Unit*), um IED (*Intelligent Electronic Device*), um sistema de controle de uma subestação, um centro de controle etc. A comunicação com a unidade remota pode ser com vários protocolos padrão (por exemplo, IEC 61870, IEC 61850) ou protocolos não padrão (por exemplo, DNP, RP570 etc.). Uma unidade remota contém pontos de dados remotos que podem ser processados por telemetria, coletados ou calculados. A classe RemoteUnit herda características de PowerSystemResource (Pacote *Core*). O propósito da classe é permitir medições das unidades remotas, e essas medições podem ser utilizadas para modelar o estado (*status*) da unidade como operacional, fora de serviço, falha da unidade etc. O atributo da classe é:

- **remoteUnitType**: indica o tipo da unidade remota {*RTU*, *SubstationControlSystem*, *ControlCenter*}.

ANEXO B – TECNOLOGIA DA INFORMAÇÃO ENVOLVIDA

B.1. UML – Unified Modeling Language [13], [22], [23]

B.1.1. Definição

UML – são as iniciais de *Unified Modeling Language*, que em português significam Linguagem Unificada de Modelagem, é uma linguagem gráfica padrão para a elaboração da estrutura de projetos complexos de *software*. A UML pode ser empregada para visualizar, especificar, construir e documentar sistemas de *software*.

A UML é o resultado da unificação das linguagens de modelagem de objetos de 3 métodos líderes do mercado: *Booch*, *Object Modeling Technique* (OMT) e *Objected-Oriented Software Engineering* (OOSE). Em 1997, a primeira versão da UML foi adotada pela OMG – *Object Management Group* – e desde então tornou-se o padrão da indústria de *software* para a modelagem de objetos e componentes.

B.1.2. Objetivos

1. Modelar sistemas (não apenas de programas) usando os conceitos de orientação a objetos;
2. Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis;
3. Criar uma linguagem de modelagem utilizável tanto pelo homem quanto pela máquina.

B.1.3. A Guerra dos Métodos

A UML acabou com a chamada “guerra de métodos” na comunidade de desenvolvedores da orientação a objeto. Orientação a objeto foi inicialmente criada pela linguagem de

programação SIMULA, mas não se popularizou até o final dos anos 80 com o advento das linguagens de programação C++ e Smalltalk. Quando a programação orientada a objetos tornou-se um sucesso, a necessidade de métodos para suportar o desenvolvimento de sistemas se seguiu. Alguns dos métodos de orientação a objeto que se popularizaram no início dos anos 90 foram: o Booch de Grady Booch, o OOSE (*Object-Oriented Software Engineering*) de Ivar Jacobson, e o OMT (*Object Modeling Technique*) de James Rumbaugh. Entre outros métodos importantes, encontravam-se *Fusion*, *Shlaer-Mellor* e *Coad-Yourdon*. Todos eram métodos completos, apesar de cada um conter pontos fortes e fracos.

A UML é a unificação das seguintes tecnologias:

- Booch – O método de Grady Booch para desenvolvimento orientado a objetos está disponível em muitas versões. Booch definiu a noção de que um sistema é analisado a partir de um número de visões, onde cada visão é descrita por um número de modelos e diagramas. O Método de Booch trazia uma simbologia complexa de ser desenhada a mão, continha também o processo pelo qual sistemas são analisados por macro e micro visões.
- OMT – Técnica de Modelagem de Objetos (*Object Modeling Technique*) é um método desenvolvido pela GE (*General Electric*) onde James Rumbaugh trabalhava. O método é especialmente voltado para o teste dos modelos, baseado nas especificações da análise de requisitos do sistema. O modelo total do sistema baseado no método OMT é composto pela junção dos modelos de objetos, funcional e casos de uso (*use-cases*).
- OOSE/Objectory – Os métodos OOSE e o Objectory foram desenvolvidos baseados no mesmo ponto de vista formado por Ivar Jacobson. O método OOSE é a visão de Jacobson de um método orientado a objetos, já o Objectory é usado para a construção de sistemas

tão diversos quanto eles forem. Ambos os métodos são baseados na utilização de casos de uso (*use-cases*), que definem os requisitos iniciais do sistema, vistos por um ator externo. O método Objectory também foi adaptado para a engenharia de negócios, onde é usado para modelar e melhorar os processos envolvidos no funcionamento de empresas.

Em termos simples, o método Booch era expressivo principalmente durante as fases de projeto e construção de sistemas, o OMT-2 era mais útil para análise e sistemas de informações com uso intensivo de dados, o OOSE fornecia excelente suporte para os casos como uma maneira de controlar a captura de requisitos, a análise e o projeto em alto nível.

A UML padronizou a modelagem orientada a objetos de uma forma que qualquer sistema, seja qual for o tipo, possa ser modelado corretamente, com consistência, fácil de se comunicar com outras aplicações, simples de ser atualizado e compreensível.

B.1.4. Histórico

Grady Booch e James Rumbaugh na *Rational Software Corporation* iniciaram os trabalhos na UML em 1994. Suas metas eram a criação de um novo método, um “Método Unificado” que iria unir os métodos Booch e OMT-2, onde Rumbaugh era o principal desenvolvedor.

Em 1995 Ivar Jacobson – o homem por trás dos métodos OOSE e Objectory – uniu-se ao grupo. A *Rational Software* ainda adquiriu a *Objective Systems*, a companhia suíça que desenvolvia e distribuía o método Objectory. Neste ponto, os futuros desenvolvedores da UML perceberam que seus trabalhos estavam mais direcionados à criação de uma linguagem padrão de modelagem e renomearam seu trabalho para "Linguagem Unificada de Modelagem".

Booch, Rumbaugh e Jacobson lançaram algumas versões preliminares da UML à comunidade de orientação a objetos. As respostas em retorno possibilitaram a eles várias idéias e sugestões a serem incorporadas a fim de aprimorar a linguagem. A versão 1.0 da UML foi lançada em Janeiro de 1997.

Ainda que as partes principais da UML sejam baseadas nos conceitos dos métodos de Booch, OMT e OOSE, os desenvolvedores ainda incluíram conceitos de outros métodos.

B.1.5. Aceitação da UML

Para estabelecer a UML, os desenvolvedores e a *Rational* perceberam que a linguagem teria que estar disponível para todos. Consequentemente, a linguagem não é proprietária e as companhias são livres para utilizá-la com seus próprios métodos.

Empresas de *software* são livres para criarem ferramentas que facilitem a utilização da UML e autores são encorajados a escrever sobre o assunto.

Durante 1996, algumas companhias associaram-se à *Rational* para formar um consórcio de parceiros da UML. Essas organizações reconheceram a UML como estratégica para seus negócios e contribuíram para a definição da UML 1.0. As diferentes companhias até janeiro de 1997 eram: *Digital Equipment Corporation*, *Hewlett-Packard*, *I-Logix*, *Intellicorp*, *IBM*, *ICON Computing*, *MCI Systemhouse*, *Microsoft*, *Oracle*, *Texas Instruments* e *Unisys*. Estas companhias estão também dando suporte na proposta de adaptar a UML como padrão para linguagens de modelagem no OMG - Grupo de Gerenciamento de Objetos.

B.1.6. Padronização OMG

Entre janeiro e julho de 1997, o grupo original de parceiros se expandiu, passando a incluir *Andersen Consulting*, *Ericsson*, *ObjectTime Limited*, *Platinum Technology*, *PTech*, *Reich Technologies*, *Softeam*, *Sterling Software* e *Taskon*. Um grupo de tarefas em semântica foi formado, liderado por Cris Kobryn da *MCI Systemhouse* e administrado por Ed Eykholt da *Rational*, com o propósito de formalizar a especificação da UML e de integrar a linguagem a outros esforços de padronização. Uma versão revisada da UML (versão 1.1) foi oferecida para padronização ao OMG em julho de 1997. Em setembro do mesmo ano, essa versão foi aceita pela ADTF (*Analysis and Design Task Force*) e pelo *Architecture Board* do OMG e, a seguir, submetida ao voto de todos os membros do OMG. A UML 1.1 foi adotada pelo OMG em 14 de novembro de 1997.

A manutenção da UML foi então assumida pela RTF (*Revision Task Force*) do OMG, sob a responsabilidade de Cris Kobryn. A RTF lançou uma revisão editorial, a UML 1.2, em junho de 1998. No final do mesmo ano, a RTF lançou a UML 1.3, em setembro de 2001 estava disponível a UML 1.4 e atualmente a versão oficial corrente é a UML 1.5, disponível desde março de 2003.

B.1.7. Aplicabilidade

A UML é utilizada para modelar sistemas, que podem possuir uma diversidade muito grande. Pode ser usada também em diferentes estágios de desenvolvimento de um sistema, desde a especificação dos requerimentos até os testes de um sistema finalizado.

Diferentes Tipos de Sistemas

A meta da UML é descrever qualquer tipo de sistema, em termos de diagramas de orientação a objeto. O uso mais comum é a criação de modelos de sistemas de *software*, mas a UML pode também ser utilizada para descrever sistemas mecânicos sem qualquer *software*, ou a organização de um negócio. Apresentamos a seguir, alguns diferentes tipos de sistemas e suas principais características:

Sistemas de Informação: Um conjunto de componentes inter-relacionados que coleta, recupera, processa, armazena e distribui informações que darão suporte ao controle e tomada de decisão de uma organização. Trabalha com grandes quantidades de dados relacionados de forma complexa, que são armazenados em bancos de dados.

Sistemas Técnicos: Trabalham e controlam equipamentos técnicos, como telecomunicações, sistemas militares ou processos industriais. Precisam manipular interfaces específicas dos equipamentos e possuem menos *softwares* padronizados do que os sistemas de informação. São freqüentemente sistemas em tempo real.

Sistemas Residentes em Tempo Real: Executados em dispositivos presentes em outros equipamentos, como telefones celulares, carros, sistemas de segurança (alarmes), etc. Isto é realizado através de programação em baixo nível que exigem suporte em tempo real.

Sistemas Distribuídos: São distribuídos em várias máquinas, onde a transferência de dados é feita facilmente de uma máquina para outra. Requerem mecanismos de comunicação sincronizada e que assegurem a integridade dos dados.

Sistemas de Software: Definem a infra-estrutura técnica que outros programas utilizam. Sistemas operacionais, bancos de dados, e interfaces de usuário que executam operações de

baixo nível no *hardware*, enquanto disponibilizam interfaces genéricas de uso de outros programas.

Sistemas de Negócios: Descrevem as metas, os recursos (humanos, financeiros, etc.), as legislações (leis, políticas da empresa, estratégias, etc.), e o trabalho atual no negócio (processos do negócio).

É importante enfatizar que a maioria dos sistemas não encaixa exatamente em alguma destas categorias, mas pertencem a mais de uma, como uma combinação destas. Por exemplo, muitos dos sistemas de informação possuem características de sistemas distribuídos e de tempo real. A UML possui capacidade de modelar todos estes tipos de sistemas.

B.1.8. Arquitetura de um Sistema

Um sistema complexo de *software* requer sua visualização sob diferentes perspectivas. Diferentes participantes – usuários finais, analistas, desenvolvedores, integradores de sistemas, técnicos, gerentes de projetos e outros envolvidos – trazem contribuições próprias ao projeto e visualiza o sistema de forma distinta em momentos diferentes ao longo deste desenvolvimento. No gerenciamento desses diversos pontos de vista, objetivando um controle do desenvolvimento iterativo e incremental, é importante considerar a arquitetura do sistema.

A arquitetura de software não está apenas relacionada à estrutura e ao comportamento, mas também ao uso, à abrangência, a adequações e a restrições de caráter econômico e tecnológico, além de questões estéticas.

A arquitetura de um sistema complexo de *software* pode ser descrita por visões.

As visões mostram diferentes aspectos do sistema que está sendo modelado. A visão não é um gráfico, mas uma abstração consistindo em uma série de diagramas. Definindo um número de visões, cada uma mostrará aspectos particulares do sistema, dando enfoque a ângulos e níveis de abstrações diferentes e uma figura completa do sistema poderá ser construída. As visões também podem servir de ligação entre a linguagem de modelagem e o método ou processo de desenvolvimento escolhido.

A Figura 48, mostra que a arquitetura de um sistema complexo de software pode ser descrita mais apropriadamente por cinco visões interligadas. Cada visão é descrita por um número de diagramas que contém informações que dão ênfase aos aspectos particulares do sistema.

Figura 48. A modelagem da arquitetura de um sistema - adaptada de [13]

Visão de Caso de Uso (*use-case*): Descreve a funcionalidade e comportamento do sistema visualizada pelos seus atores externos (usuários finais, analistas, desenvolvedores e pessoal de teste). Essa visão é central, e seu conteúdo é base do desenvolvimento de outras visões do sistema. Com a UML, os aspectos estáticos dessa visão são capturados em diagramas de caso de uso, enquanto aspectos dinâmicos são capturados em diagramas de interação (diagramas de sequência e diagramas de colaboração), diagramas de gráfico de estado e diagramas de atividades.

Visão de Projeto: Descreve como a funcionalidade do sistema será implementada. É executada principalmente pelos analistas e desenvolvedores. Em contraste com a Visão de Caso de Uso, a Visão do Projeto observa e estuda o sistema internamente. Ela descreve e especifica a estrutura estática do sistema (classes, objetos e relacionamentos) e as colaborações dinâmicas quando os objetos enviarem mensagens uns para os outros para realizarem as funções do sistema. Propriedades como persistência e concorrência são definidas nesta fase, bem como as interfaces e as estruturas de classes. Com a UML, os aspectos estáticos dessa visão são captados em diagramas de classes e de objetos; os aspectos dinâmicos são captados em diagramas de interações (diagramas de seqüência e diagramas de colaboração), diagramas de gráfico de estados e diagramas de atividades.

Visão de Processo: Trata a divisão do sistema em processos e processadores. Este aspecto, que é uma propriedade não funcional do sistema, permite uma melhor utilização do ambiente onde o sistema será implantado, se o mesmo possui execuções paralelas, e se existe dentro do sistema um gerenciamento de eventos assíncronos. Uma vez dividido o sistema em linhas de execução de processos concorrentes (*threads*), esta visão do processo deverá mostrar como se dá a comunicação e a concorrência destas *threads* – cuida principalmente de questões referentes ao desempenho, à *escalabilidade* e ao *throughput* do sistema. Com a UML, os aspectos estáticos e dinâmicos dessa visão são captados nos mesmos tipos de diagramas da visão de projeto, mas com o foco voltado para as classes ativas que representam essas *threads* e processos.

Visão de Implementação: mostra a organização física do sistema, os computadores, os periféricos e como eles se conectam entre si. Esta visão será executada pelos desenvolvedores, integradores e pessoal de teste. Com a UML, os aspectos estáticos dessa visão são capturados

em diagramas de componentes; os aspectos dinâmicos são capturados em diagramas de interações (diagramas de sequência e diagramas de colaboração), de gráficos de estados e de atividades.

Visão de Implantação: abrange os nós que formam a topologia de *hardware* em que o sistema é executado. Essa visão direciona principalmente a distribuição, o fornecimento e a instalação das partes que constituem o sistema físico. Com a UML, os aspectos estáticos dessa visão são capturados em diagramas de implantação; os aspectos dinâmicos são capturados em diagramas de interações (diagramas de sequência e diagramas de colaboração), diagramas de gráfico de estados e diagramas de atividades.

B.1.9. Desenvolvimento de Sistemas: Visão Tradicional ou Orientada a Objetos

Conforme [13], (BOOCH; RUMBAUGH; JACOBSON, 2000) comentam que

[...] A visão tradicional no desenvolvimento de softwares adota a perspectiva de um algoritmo. Nessa visão, o principal bloco de construção do software é o procedimento ou a função. Essa perspectiva conduz os desenvolvedores a voltar seu foco de atenção para questões referentes ao controle e à decomposição de algoritmos maiores em outros menores. Não existe nenhuma grande desvantagem nessa solução, com exceção da tendência a permitir sistemas instáveis. À medida que os requisitos se modificam (e isso certamente ocorrerá) e o sistema cresce (o que também acontecerá), será difícil fazer a manutenção de sistemas construídos a partir do foco em algoritmos.

A visão contemporânea no desenvolvimento de software adota uma perspectiva orientada a objetos. Nessa visão, o principal bloco de construção de todos os sistemas de software é o objeto ou a classe. Explicando de uma maneira simples, um objeto é alguma coisa geralmente estruturada a partir do vocabulário do espaço do problema ou do espaço da solução; uma classe é a descrição de um conjunto de objetos comuns. Todos os objetos têm uma identidade (você pode atribuir-lhes nomes ou diferenciá-los dos demais objetos de alguma maneira), um estado (costuma haver dados a eles associados) e um comportamento (você poderá fazer algo com o objeto ou ele poderá fazer algo com outros objetos).

A Figura 49, mostra graficamente a diferença entre a visão tradicional de desenvolvimento de sistemas (enfoque em sistemas) e a visão contemporânea (enfoque em objetos).

Figura 49. Desenvolvimento de sistemas: Visão tradicional e Visão contemporânea

A vantagens da orientação a objetos são:

- Reutilização dos objetos: Os dados e os processos são manipulados por objetos, e não limitados apenas dentro dos programas, facilitando a utilização dos objetos construídos em diferentes sistemas.
- Modularidade: O sistema formado por objetos e não por programas, facilita o trabalho futuro de manutenção desses objetos.
- Conceitos da realidade: A orientação a objetos baseia-se em conceitos da realidade – objetos, estados e estímulos – para a modelagem de sistemas de *software*, tornando seu desenvolvimento mais simples e natural.

B.1.10. Fases de Desenvolvimento de Sistemas

Existem cinco fases no desenvolvimento de sistemas: Análise de Requisitos, Análise, Projeto (*Design*), Programação e Testes. As fases devem ser executadas concomitantemente de forma

que problemas detectados numa certa fase modifiquem e melhorem as fases desenvolvidas ou em desenvolvimento, no intuito que o resultado final gere um produto de alta qualidade e desempenho.

Análise de Requisitos

A UML possui casos de uso para retratar os requisitos do cliente. Um caso de uso é uma descrição de uma funcionalidade (uma utilização específica do sistema) que o sistema fornece. Através da modelagem de casos de uso, os atores internos que têm interesse no sistema são modelados de acordo com a funcionalidade que eles requerem do sistema. Os atores e casos de uso são modelados com relacionamentos, e tem suas associações de comunicações um com o outro divididas em hierarquias. Os atores e os casos de uso são descritos na forma de um diagrama de casos de uso na UML. Cada caso de uso modelado é descrito através de um texto e especifica os requisitos do cliente, neste momento, não é considerada como a funcionalidade será implementada. Uma análise de requisitos pode também ser realizada com processos de negócio, não apenas para sistemas de software.

Análise

O estágio de análise está relacionado com as abstrações primárias (classes e objetos) e mecanismos que estão presentes no domínio do problema. As classes que modelam estas abstrações e mecanismos são identificadas, juntamente com seus relacionamentos umas com as outras, e descritas num diagrama de classe na UML. Colaborações entre as classes para desempenhar casos de uso são também descritas, através de qualquer um dos diagramas de comportamento da UML. Na análise, somente as classes que estão no domínio do problema (conceitos do mundo real) são modelados – e não as classes técnicas que definem detalhes e

soluções no sistema do software, como as classes de interface com o usuário, bancos de dados, comunicações, e assim por diante.

Projeto (*Design*)

Na fase de projeto, o resultado da análise é expandido em soluções técnicas. Novas classes são adicionadas de forma a suportar a infra-estrutura técnica: a interface do usuário e de periféricos, gerenciamento de bancos de dados, comunicação com outros sistemas, dentre outros. As classes do domínio do problema, modelados na fase de análise, são incorporados nessa infra-estrutura técnica, tornando possível modificar tanto o domínio do problema quanto à infra-estrutura. A fase de projeto resulta em uma especificação detalhada para a fase de programação do sistema.

Programação

Na fase de programação, as classes definidas no estágio de projeto são convertidas em código utilizando alguma linguagem de programação orientada a objeto, sendo que a utilização de uma linguagem procedural não é recomendada.

Testes

Um sistema é normalmente validado em testes de unidade, integração e aceitação. Os testes de unidade são para classes individuais ou um grupo de classes e são tipicamente realizadas pelo programador. Os testes de integração são aplicados utilizando as classes e componentes integrados, buscando verificar se as classes cooperam uma com as outras como especificado nos modelos. Os testes de aceitação verificam as funcionalidades do sistema conforme especificado nos primeiros diagramas de casos de uso (*use-cases*).

B.1.11. Elementos

Os elementos da UML são as simbologias e notações utilizadas para construir os modelos orientados a objetos. Os principais elementos são os blocos de construção.

B.1.12. Blocos de Construção

Os blocos de construção da UML são os insumos utilizados pelos diagramas para representar partes de um sistema de *software*.

Os blocos de construção existentes na UML, são: Itens, Relacionamentos e Diagramas.

B.1.13. Itens

Os itens são os blocos de construção da modelagem orientada a objetos. Existem quatro tipos de itens na UML. São eles:

1. Itens estáticos: Representam a estrutura do modelo, elementos conceituais e físicos (classes, colaborações, casos de uso, etc.). Retratar o esqueleto e a estrutura estável de um sistema.
 - Classe: É um conjunto de objetos que compartilham os mesmos atributos, operações e relacionamentos.

A Figura 50, mostra as representações gráficas de uma Classe e a Figura 51, mostra o exemplo da Classe Cliente.

Figura 50. Representações gráficas de uma Classe - [22]

Figura 51. Exemplo da Classe Cliente - adaptada [22]

- Interface: É uma coleção de operações que especificam serviços de uma classe ou componente; descreve o comportamento externamente visível desse elemento. A interface é representada graficamente como um círculo e o respectivo nome, conforme mostra a Figura 52.

Figura 52. Representação gráfica de uma Interface - [13]

- Colaboração: É o nome dado à interação entre duas ou mais classes, com o objetivo de fornecer algum comportamento cooperativo. Ocorrem por meio da troca de mensagens entre dois ou mais objetos.

A representação gráfica de uma Colaboração é mostrada na Figura 53.

Figura 53. Representação gráfica de uma Colaboração - [13]

- Caso de uso: É um documento que descreve os cenários pretendidos para um sistema, com o objetivo de atender às necessidades do usuário. É representado graficamente, conforme demonstrado na Figura 54.

Figura 54. Representação gráfica de um Caso de Uso - [13]

- Componente: É a parte física de um sistema. Existem diferentes tipos de componentes, como os componentes COM+ e Java Beans. A Figura 55, mostra a representação gráfica de um componente.

Figura 55. Representação gráfica de um Componente - adaptada [13]

- Nó: É um elemento físico existente em tempo de execução que representa um recurso computacional.

A Figura 56, mostra a representação gráfica de um nó.

Figura 56. Representação gráfica de um Nó - [13]

2. Itens dinâmicos: Definem o comportamento do modelo (interações e estados dos objetos). Representam as partes de um sistema que possam ter alguma alteração. Existem dois tipos principais de itens dinâmicos. São eles:

- Interação: Representa a realização dos processos de um sistema, por meio da troca de mensagens entre os objetos. Pode envolver mensagens, seqüências de ações e conexões entre os objetos. Uma interação é representada por uma flecha cheia, conforme mostrado na Figura 57.

Figura 57. Representação gráfica de uma Interação - [13]

- Máquina de estados: Representa um comportamento que especifica as seqüências de estados pelos quais os objetos passam durante sua existência, de acordo com os eventos atendidos. A Figura 58, mostra a representação de um Estado.

Figura 58. Representação gráfica de um Estado - [13]

3. Pacotes: Organizam os modelos criados na UML. Todos os blocos de construção criados na UML (casos de uso, classes, componentes, etc.), bem como os diagramas que tratam desses blocos, podem ser organizados em pacotes.

Deve-se fazer essa organização observando a semelhança de cada bloco e diagrama da UML, no que diz respeito à solução do problema que cada um está empregando. A Figura 59, mostra a representação gráfica de um Pacote.

Figura 59. Representação gráfica de um Pacote - [13]

4. Notas: São as partes explicativas do modelo na UML. Representam comentários, observações e esclarecimentos que podem ser utilizados para qualquer elemento da UML. A Figura 60, mostra a representação gráfica de uma Nota.

Figura 60. Representação gráfica de uma Nota - [22]

B.1.14. Relacionamentos

Os relacionamentos reúnem os itens criados na UML. Existem seis tipos de relacionamentos:

1. Dependência: É uma conexão semântica entre dois itens, um independente e outro dependente. Uma mudança no elemento independente pode afetar a semântica do

elemento dependente. A Figura 61, mostra a representação gráfica de um Relacionamento de Dependência.

Figura 61. Representação gráfica de um Relacionamento de Dependência - [13]

2. Associação: Descreve relacionamentos entre objetos. Cada associação tem duas pontas, onde em cada ponta está ligado a um objeto. Por exemplo, poderíamos ter um relacionamento de associação entre o objeto *Empregador* na primeira ponta, e o objeto *Funcionário* na outra ponta. A Figura 62, mostra a representação gráfica de um Relacionamento de Associação.

Figura 62. Representação gráfica de um Relacionamento de Associação - [13]

Multiplicidade de associação: É a indicação de quantos objetos podem participar de determinado relacionamento. Observando a Figura 62, pode-se dizer que o objeto *Empregador* pode ter muitos *Funcionários*, enquanto esse objeto pode ter zero ou um objeto *Empregador*.

Em UML, temos basicamente os seguintes tipos de multiciplidades: “0” (zero), “1” (um), “0..1” (zero ou um), “0..*” (zero ou mais), “1..*” (um ou mais) e “*”. (muitos).

3. Agregação: É um tipo de associação na qual o todo é relacionado com suas partes (um objeto todo contém os objetos-partes – relacionamento todo/parte). É representada como uma linha de associação com um diamante junto à classe agregadora. A Figura 63, mostra a representação gráfica de um Relacionamento de Agregação.

Figura 63. Representação gráfica de um Relacionamento de Agregação - [22]

4. Composição: É um tipo de relacionamento muito semelhante à agregação. Em uma composição, o objeto-parte pode pertencer somente ao objeto-todo. Além disso, geralmente o objeto-todo vive e morre com suas partes, isto é, acontece uma remoção em cascata. É representada como uma linha de associação com um diamante preto, junto à classe agregadora. A Figura 64, mostra a representação gráfica de um Relacionamento de Composição.

Figura 64. Representação gráfica de um Relacionamento de Composição - [22]

5. Generalização/Especialização: A generalização é a capacidade de se criar supertipos que encapsulam a estrutura e o comportamento comum a vários subtipos. Os procedimentos para se obter a generalização são:
- Identificar similaridades de estrutura/comportamento entre várias classes.
 - Criar o supertipo para encapsular a estrutura e o comportamento comum a mais de uma classe.
 - Os objetos originais passam a ser subtipos do novo supertipo criado.

A especialização, é a capacidade de se criar subtipos que representam refinamentos nos quais a estrutura e/ou comportamento do supertipo são adicionados ou modificados. Os procedimentos para obter a especialização são:

- Perceber que algumas classes apresentam estrutura e/ou comportamento especializado.
- Criar subtipos de acordo com sua especialização.

A Figura 65, mostra a representação gráfica de um Relacionamento de Generalização/Especialização.

Figura 65. Representação gráfica de um Relacionamento de Generalização - [22]

6. Realização: É um relacionamento entre itens, em que um item implementa comportamentos especificados por outros. Pode ser realizado entre interfaces e classes ou componentes que as realizam; e entre casos de uso e as colaborações que os realizam. É representado por uma seta pontilhada, como mostrado na Figura 66.

Figura 66. Representação gráfica de um Relacionamento de Realização - [13]

B.1.15. Diagramas

Um diagrama é a apresentação gráfica de um conjunto de elementos – geralmente representados como gráficos de vértices (itens) e arcos (relacionamentos) – que são desenhados para permitir a visualização de um sistema sob diferentes perspectivas. O diagrama representa uma visão parcial dos elementos que compõem o sistema. O mesmo elemento pode ou não aparecer em todos os diagramas e um diagrama pode conter qualquer combinação de itens e relacionamentos.

A UML define nove tipos de diagramas, divididos em duas categorias: quatro tipos de diagramas representam a estrutura estática da aplicação e cinco representam diferentes aspectos de comportamento dinâmico.

Diagramas Estruturais (*Structural Diagrams*)

- Diagrama de Classes (*Class Diagram*): Demonstra a estrutura estática das classes de um sistema, que representam o “vocabulário do sistema”, que é gerenciada pela aplicação modelada. Classes podem se relacionar com outras classes de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é especialização de outra classe), ou em pacotes (classes agrupadas por características similares). Todos estes relacionamentos são mostrados no diagrama de classes juntamente com suas estruturas internas, que são os atributos e operações. O diagrama de classes é considerado estático visto que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema. Um sistema normalmente possui alguns diagramas de classes, pois nem todas as classes estão inseridas em um único diagrama e uma classe pode participar de vários diagramas de classes. Os diagramas de classes servem de base para os diagramas de componentes e de implantação, são importantes para a visualização, especificação e documentação de modelos estruturais, como também para a construção de sistemas executáveis por intermédio de engenharia de produção e reversa.

A Figura 67, mostra o exemplo de um Diagrama de Classes.

Figura 67. Exemplo de um Diagrama de Classes - [13]

- Diagrama de Objetos (*Object Diagram*): Faz a modelagem de instâncias de itens contidos nos diagramas de classes e utiliza quase a mesma notação. O diagrama de objetos é como se fosse o perfil do sistema num determinado instante de sua execução. A mesma notação de classes é utilizada com 2 exceções: os objetos são escritos com seus nomes sublinhados e todas as instâncias num relacionamento são mostradas. Os diagramas de objetos não são tão importantes como os diagramas de classes, mas são úteis para exemplificar diagramas complexos de classes auxiliando sua compreensão. Diagramas de objetos também são

utilizados como parte dos diagramas de colaboração, onde a colaboração dinâmica entre os objetos do sistema são mostrados.

A Figura 68, mostra um exemplo do Diagrama de Objetos.

Figura 68. Exemplo de um Diagrama de Objetos - [13]

- Diagrama de Componentes (*Component Diagram*): O diagrama de componentes, tal como o diagrama de implantação, mostra o sistema por um lado funcional, expondo as relações entre seus componentes e a organização de seus módulos durante sua execução.

O diagrama de componentes descreve os componentes de *software* e suas dependências entre si, representando a estrutura do código gerado. Os componentes são a implementação, na arquitetura física, dos conceitos e da funcionalidade definidos na arquitetura lógica (classes, objetos e seus relacionamentos). Eles são tipicamente os executáveis, tabelas, arquivos e documentos implementados no ambiente de desenvolvimento.

Um componente é mostrado em UML como um retângulo e dois retângulos menores do seu lado esquerdo. O nome do componente é escrito abaixo ou dentro de seu símbolo. A Figura 69, mostra um exemplo do Diagrama de Componentes.

Figura 69. Exemplo de um Diagrama de Componentes - adaptada [13]

- Diagrama de Implantação (*Deployment Diagram*): Demonstra a arquitetura em tempo de execução dos processadores (nós), componentes físicos (*devices*) e *softwares* que são executados no ambiente onde o sistema desenvolvido será utilizado.

O diagrama de implantação mostra a arquitetura física do *hardware* e *software* do sistema. Pode mostrar os computadores (nós) e periféricos, juntamente com os tipos de conexões que eles estabelecem entre si. Está relacionado aos diagramas de componentes, pois tipicamente um nó inclui um ou mais componentes.

A Figura 70, mostra um exemplo do Diagrama de Implantação.

Figura 70. Exemplo de um Diagrama de Implantação - [13]

Diagramas de Comportamento (*Behavior Diagrams*)

- Diagrama de Caso de Uso (*Use-Case Diagram* – usado por algumas metodologias durante a fase de análise de requisitos): Representa um conjunto de atores, casos de uso e os relacionamentos entre eles. Atores, são papéis desempenhados por qualquer usuário de um caso de uso, é quem solicita os serviços disponíveis em casos de uso.

Um ator pode ser uma pessoa ou um *hardware* que interage com o sistema ou outro sistema que tenha a necessidade de utilizar o caso de uso.

A Figura 71, mostra um exemplo do Diagrama de Caso de Uso.

Figura 71. Exemplo de um Diagrama de Caso de Uso - [13]

- Diagrama de Atividade (Activity Diagram): Representa a modelagem do fluxo de controle de uma atividade para uma outra no sistema. São exibidos os estados das atividades e ações, transições e objetos. A Figura 72, mostra um exemplo do Diagrama de Atividade.

Figura 72. Exemplo de um Diagrama de Atividade - [13]

– Diagramas de Interações

- Diagrama de Seqüência (*Sequence Diagram*): Enfatiza à ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Mensagens, são os serviços solicitados por um objeto a outro, e as respostas devolvidas a essas solicitações. A Figura 73, mostra um exemplo do Diagrama de Seqüência.

Figura 73. Exemplo de um Diagrama de Seqüência - [13]

- Diagrama de Colaboração (*Collaboration Diagram*): Enfatiza à ordenação estrutural em que as mensagens são trocadas entre os objetos de um sistema. A Figura 74, mostra um exemplo do Diagrama de Colaboração.

Figura 74. Exemplo de um Diagrama de Colaboração - [13]

- Diagrama de Gráfico de Estado (Statechart Diagram): Representa os estados possíveis de um objeto em particular. São exibidos os estados de um objeto, eventos, transições e atividades. Pode ser usado principalmente para a modelagem de estados de classes e colaborações. A Figura 75, mostra um exemplo do Diagrama de Gráfico de Estado.

Figura 75. Exemplo de um Diagrama de Gráfico de Estado - [13]

B.1.16. Visão Geral dos Diagramas

Os diagramas da UML representam diferentes partes do modelo de uma sistema. A Figura 76, demonstra como os diagramas se relacionam, formando toda a modelagem orientada a objetos.

Figura 76. Relacionamento entre os Diagramas da UML

B.2. Banco de Dados

Conforme autorização do próprio autor, este anexo é reprodução de dois tópicos da sua Tese de Doutorado [24].

B.2.1. Sistemas Gerenciadores de Bancos de Dados

O conceito de sistemas Gerenciadores de Bancos de Dados (SGBD) tem origem na década de 70, objetivando maior controle sobre a informação e promover a redução da duplicação de dados, através de seu compartilhamento por múltiplos programas.

Um sistema de banco de dados pode ser entendido como um conjunto de dispositivos orientados ao armazenamento de informações, composto de um conjunto de informações organizadas segundo critérios e um conjunto de programas que respondem pela inserção, recuperação e manutenção dos dados. Esse sistema, tanto pode ser utilizado apenas como um meio conveniente de organizar informações relevantes de um usuário, como para armazenar todos os dados de uma grande empresa, suportando acesso concorrente de inúmeros usuários.

O ponto chave desse conceito é a capacidade de separar o processo (*software*) dos dados, eliminando assim, a necessidade de cada aplicação manter seu próprio conjunto particular de dados, muitas vezes redundantes com outras aplicações.

Dentro desse conceito, podemos classificar as funções básicas de um SGBD em:

- **Manipulação de dados:** manutenção dos dados através de procedimentos de inserção, atualização, remoção e recuperação.
- **Definição de dados:** estruturar os elementos dos dados em esquemas lógicos e físicos.

- **Restrições de integridade:** garantir a segurança, integridade e concorrência dos dados.

No sentido de prover essas necessidades básicas, um SGBD deverá oferecer, pelo menos, o seguinte conjunto de recursos:

- linguagem de manipulação de dados;
- linguagem de definição de dados;
- ferramentas de população da base de dados;
- ferramentas para recuperação e “backup” de dados;
- interface de programação de banco de dados (API);
- ferramentas de desenvolvimento de aplicações;
- ferramentas de questionamento (“query”) e relatórios.

B.2.2. Modelos de Dados

A arquitetura para armazenamento das informações em SGBD é estabelecida a partir de um modelo de dados, que é uma forma de representação resultante de uma abstração do mundo real. Os SGBDs comerciais disponíveis no mercado geralmente adotam o modelo hierárquico, modelo de rede, modelo relacional, ou ainda orientado a objetos, sendo que atualmente o modelo relacional é o mais difundido.

O modelo hierárquico oferece ao usuário uma visão do banco de dados, como uma estrutura de árvores, onde os registros se equivalem às folhas enquanto que as ligações se equivalem aos ramos. Assim sendo, cada registro pode ter um número qualquer de descendentes, mas

apenas um ascendente (exceto a raiz, que não possui ascendentes). Um dos primeiros SGBDs comercializados que suportavam esse modelo foi o IMS (*“Information Management System”*) da IBM, disponibilizado comercialmente a partir de 1968.

Já o modelo de rede proporciona ao usuário uma visão análoga à de um grafo ou de uma malha de ligações entre registros, do tipo de um para muitos; ou seja, um registro pode estar envolvido em mais de um relacionamento, podendo apresentar vários ascendentes e vários descendentes. Um dos principais produtos que adota essa modelagem é o ADABAS C, que ainda é utilizado pela maior parte das empresas de distribuição de energia elétrica, no Brasil.

As bases de dados relacionais baseiam-se no conceito que os dados são vistos pelo usuário como se estivessem dispostos em estruturas na forma de tabelas arranjadas em linhas (ou tuplas) e colunas (ou atributos ou domínio). Cada tabela armazena os dados relevantes a um determinado tipo de entidade, enquanto as colunas representam os atributos dessa tabela. Cada coluna contém dados de um mesmo tipo. A interseção de uma coluna e uma linha de uma tabela, sempre produz um único valor e nunca uma outra tabela.

As operações entre tabelas são ditadas pela álgebra relacional, que oferece as seguintes operações: união, interseção, diferença, produto cartesiano, projeção, seleção ou restrição, junção e divisão.

O modelo relacional não determina a forma como as tabelas são representadas fisicamente mas, de modo geral, cada tabela é salva num arquivo fisicamente separado, cujos registros representam as linhas da tabela, enquanto as colunas são representadas pelos campos do registro.

O termo relacional é derivado da teoria dos conjuntos, onde uma relação é um tipo particular de conjunto que possui propriedades similares às apresentadas pelas colunas das tabelas. Dentre os produtos disponíveis no mercado podemos citar: ADABAS D, Sybase, Oracle, DB2, MS SQL Server, Ingres, MS Access, MySQL, etc.

Aplicações que requeiram muitos tipos diferentes de tuplas (registros) e necessitam de “joins” que acessem seis ou mais tipos de tuplas esgotam a capacidade do modelo relacional, quanto a desempenho e complexidade do modelo de dados, propiciaram o desenvolvimento de SGBD orientado a objetos (SGBDOO). Essa arquitetura orientada a objetos, teve início na metade da década de 80 em nível de protótipo, sendo que os primeiros produtos comerciais apareceram no início dos anos 90.

Um SGBDOO pode ser definido como sendo um SGBD que suporta diretamente os conceitos de orientação a objetos, tais como: o objeto, a mensagem, o método e a classe. Nesse paradigma um objeto representa uma entidade do mundo real, apresentando um estado interno e uma interface. A interface é composta de um conjunto de mensagens que um objeto pode trocar, podendo ser tanto uma solicitação para alteração do seu estado, como para retornar um resultado qualquer. A fim de que um objeto possa responder a uma mensagem, ele necessita executar um método apropriado, que nada mais é do que um conjunto de linhas de código escritas numa linguagem qualquer, com suporte a objetos.

Dessa forma o SGBDOO apresenta algumas características obrigatórias que são comuns as demais arquiteturas, tais como persistência, concorrências, recuperação, gerenciamento do armazenamento dos dados, além de mecanismos de “query”, possuindo ainda funções para suporte ao paradigma da orientação a objetos, tais como: abstração, herança, encapsulação e

polimorfismo. São exemplos de SGBDOO comerciais: Gemstone, O₂, Objectivity/BD, Jasmine, Versant e Object Store.

A imaturidade dos sistemas orientados a objetos aliado aos elevados preços dos SGBDOO contribuíram para o desenvolvimento de produtos híbridos, que na verdade implementam uma camada de abstração de dados sobre os métodos relacionais, permitindo o manuseio de estruturas de dados mais complexas. Esses produtos, denominados de objeto-relacionais, embora não implementem todas as funcionalidades do paradigma de OO, facilitam a migração dos sistemas procedurais para sistemas puramente objeto. São exemplos de produtos com essa característica: MS SQL Server, Oracle 8.x (ou superior) e o Universal Database (DB/2).

B.3. XML – eXtensible Markup Language

B.3.1. Panorama Predecessor

B.3.1.1. Internet

A Internet nasceu como um projeto do departamento de defesa norte americano, desenvolvido pela *Advanced Research Projects Agency* (ARPA), e por isso recebeu o nome de ARPANET. A arquitetura da ARPANET foi desenvolvida de 1959 a 1969, e tinha como objetivo principal fornecer um sistema distribuído de comunicações entre computadores que poderia sobreviver a um ataque. Nos anos 70, as universidades e outras instituições que faziam trabalhos relativos à defesa, tiveram permissão para se conectar à ARPANET. Em 1975, existiam aproximadamente 100 *sites*.

No final dos anos 1970, a ARPANET tinha crescido tanto que o seu protocolo de comutação de pacotes original, chamado de *Network Control Protocol* (NCP), tornou-se inadequado. Em

um sistema de comutação de pacotes, os dados a serem comunicados são divididos em pequenas partes. Essas partes são identificadas de forma a mostrar de onde vieram e para onde devem ir, assim como os cartões postais no sistema postal. Assim também como os cartões postais, os pacotes possuem um tamanho máximo, e não são necessariamente confiáveis.

Os pacotes são enviados de um computador para outro até alcançarem o seu destino. Se algum deles for perdido, ele poderá ser reenviado pelo emissor original. Para eliminar retransmissões desnecessárias, o destinatário confirma o recebimento dos pacotes.

Um novo protocolo chamado TCP/IP (*Transfer Control Protocol/Internet Protocol*) desenvolvido em UNIX substituiu o NCP. A maior vantagem do TCP/IP era que ele permitia, o que parecia ser impossível na época, o crescimento praticamente ilimitado da rede, além de ser fácil de implementar em computadores com arquitetura de *hardware* distinta e diferentes sistemas operacionais.

As facilidades de comunicação e troca de dados permitiram que, na metade dos anos 80, aumentasse o interesse entre pesquisadores, educadores e outros grupos, justificando a fabricação de equipamentos especificamente para a implementação da Internet.

Várias empresas começaram a se interessar pela fabricação e venda de roteadores, equipamentos necessários à conexão. A década de 90 marcou a Internet pela introdução de serviços que a tornaram mais popular, visto que eram dirigidos a um público amplo e não apenas ao meio acadêmico.

Nos últimos anos a Internet passou a estar disponível para a grande massa popular. Desde então, ela vem despertando grande interesse do público em geral. Praticamente todos os

setores da economia, desde usuários domésticos, profissionais liberais e até grandes empresas tem demonstrado forte interesse pela Internet.

O sucesso da Internet é reconhecido em qualquer segmento da atividade humana, e pode ser atribuído a inúmeras facilidades, tais como:

- Troca de informações de modo rápido e eficiente;
- Acesso a diversos tipos de especialidades;
- Atualização constante em diferentes assuntos de interesse;
- De possuir e disponibilizar o maior acervo mundial;
- Disponibilizar através das *home pages* assuntos comerciais ou pessoais;
- Trabalhar em equipe;
- Globalização.

B.3.1.2. HTML – Hiper Text Markup Language

HTML é baseada na SGML¹¹, e é abreviação de *HyperText Markup Language*. O termo *HyperText* significa textos que possuem ligações (*links*) para outros textos, e o termo *Markup* significa anotações ou marcações na estrutura de um texto.

A HTML foi a primeira linguagem de marcação para a Internet, possui uma importância significativa na evolução da *Web* e suas características e limitações provocaram o surgimento

¹¹ SGML *Standard Generalised Markup Language*, é um padrão ISO derivado da linguagem “GML” da IBM. Ela permite definir a estrutura de um documento e a relação lógica de suas partes. Esta estrutura pode ser checada e validada contra uma DTD (*Document Type Definition*). A SGML define a sintaxe para o documento, como também, a sintaxe e semântica da DTD.

de uma nova linguagem de marcação: a XML.

No início dos anos 90, Tim Berners-Lee, do *European Laboratory for Particle Physics* (mais conhecido como CERN), propôs um novo conjunto de protocolos para um sistema de distribuição de informações pela Internet. Esse protocolo ficou conhecido como WWW, sendo adotado por outras organizações. Para definir o padrão do protocolo, foi formado um consórcio chamado *W3 Consortium* (W3C), que até hoje define versões e padrões para as necessidades da *Web*.

A HTML é a linguagem de programação utilizada para criar uma página *Web* e é composta de textos e comandos especiais chamados “*tags*”, que funcionam como marcadores de texto. A HTML possui como finalidade básica formatar o texto e imagens exibidas e criar ligações entre páginas *Web*.

Como em qualquer outra linguagem, regras de sintaxe devem ser seguidas para que o interpretador da linguagem entenda corretamente o código fonte especificado. Esse código fonte é interpretado por um *Browser* (*MS Internet Explorer*, *Netscape*, *Mozilla*, etc.), que se encarregará de executar os comandos ou *tags* do código para formatar e acessar recursos disponíveis nas páginas da Internet.

Existem centenas de programas para editoração de HTML. Essa linguagem tornou-se tão popular e necessária para a *Web* que os principais *softwares* do mercado criaram ferramentas para gerar esse código.

Apesar da HTML ser chamada de linguagem de programação pela maioria dos profissionais envolvidos na área, ela é apenas uma linguagem de marcação de texto que permite especificar

como o texto será visualizado na tela do *browser*, definindo os detalhes do documento com a utilização dos marcadores (*tags*).

A Figura 77, mostra um pequeno exemplo da linguagem HTML. O texto foi simplesmente inserido no *MS Word* e salvo como um documento no formato HTML. Esta página *Web* é mostrada no *browser* juntamente com o código HTML fonte.

Todas as *tags* HTML, devem estar entre os sinais de (< e >) e devem possuir marcadores de início e fim. O documento inteiro deve ser identificado como HTML (<html>...</html>), ter uma área de cabeçalho (<head>...</head>), com o título do documento (<title>...</title>), e uma área como corpo (<body>...</body>). Quando esse documento é interpretado pelo *browser*, todos os marcadores são reconhecidos, porém ficam ocultos e não são apresentados na tela. Nos pontos demarcados do documento são aplicadas as formatações especificadas pelos marcadores.

Figura 77. Exemplo da Sintaxe HTML

Conforme [25], as principais características da HTML, são:

- Possibilita a formatação dos documentos usando vários estilos de letras, incluindo uma variedade de títulos;
- Inclui *hiperlinks* que apontam para outros documentos *Web* ou serviços em sistemas de computador por toda a Internet, permitindo desta forma a ligação entre eles;
- Possui uma variedade de recursos para criação e organização de listas;
- Recursos para criação e manipulação de tabelas e listas com tópicos;
- A capacidade de encaixar imagens gráficas diretamente em um documento HTML, que podem ser *hiperlinks* para outros documentos;

- Permite a divisão de uma imagem por meio de coordenadas, possibilitando a criação de mapas de imagens sensíveis ao “*click*” do mouse, como “*hot spots*” (pontos quentes) para acessar diferentes pontos na Internet;
- conteúdo do documento HTML é basicamente texto puro e por esse motivo pode ser criado a partir de qualquer editor de texto (*edit, notepad, Word, Wordpad...*)
- Existem conversores de vários formatos, por exemplo do “.doc” para HTML e do “.rtf” para HTML, etc.
- Possibilita a criação de páginas com várias janelas (*frames*) para melhor dividir os conteúdos a serem apresentados.

As principais deficiências da HTML, são:

- Qualidade de apresentação do documento;
- Conjunto de tags da HTML é fixo;
- As informações do documento não possuem significado;
- Os documentos HTML não são reutilizáveis.

B.3.2. Definição

A XML (*Extensible Markup Language*) é uma linguagem de marcação (*tags*), que provê um formato para descrever dados estruturados, proposta pelo Consórcio da *World Wide Web* (W3C). A XML é um subconjunto da linguagem mãe SGML, genérica, muito poderosa e de grande complexidade para ser implementada na Internet. Sendo assim, a XML é uma versão reduzida da SGML, sendo criada para ser utilizada exclusivamente na *Web* [26].

B.3.3. Histórico

Em 1996, um grupo de especialistas¹² [27] em SGML (*Standard Generalized Markup Language*), a principal linguagem de marcação da qual surgiu a HTML, se uniram para definição de um novo padrão de marcação que pudesse ser utilizado na Internet, constituindo-se em uma versão simplificada da SGML, cujo objetivo principal era fornecer aos desenvolvedores da *Web* maneiras de definir e criar seus próprios marcadores e atributos quando necessário, em vez de estarem restritos ao esquema de marcação da HTML.

Resumidamente, as características desejadas inicialmente para a XML se referiam a três partes: a definição da linguagem em si, a definição da ligação entre os documentos e a forma de apresentação dos documentos.

Conforme [25], as principais características desejáveis para a implementação da XML na *Web* eram as seguintes:

- Criar uma linguagem simples, que possibilite a rápida construção de documentos para implementação na Web;
- Fornecer suporte à criação de aplicações compatíveis com a abordagem da HTML;
- Possibilitar o desenvolvimento de uma grande variedade de aplicativos, aproveitando-se de seus recursos;
- Fornecer um mecanismo de apresentação genérico e poderoso, permitindo ao desenvolvedor criar a forma de apresentação que mais se adapte às suas necessidades;

¹² O Grupo de Especialista em SGML no intuito de gerar a especificação da XML para o W3C era composto por: Jon Bosak, C.M. Sperberg-McQueen, James Clark, Tim Bray, Steve DeRose, Eve Maler, Eliot Kimber, Dave Hollander, Makoto Murata, Peter Sharpe e Jean Paoli.

- Fornecer suporte para a criação de marcadores personalizados, definidos pelo desenvolvedor do documento *Web*;
- Permitir a criação de documentos que pudessem ser validados, isto é, que existisse uma forma de verificar a estrutura do documento, verificando se seus elementos eram válidos, da mesma forma que ocorria com a SGML;
- Fornecer suporte para a criação de *hyperlinks* que fossem compatíveis com a especificação de endereços URL (*Uniform Resource Locator*), de modo a criar ligações entre documentos.
- Fornecer um mecanismo de folha de estilo genérico e poderoso, que possibilitasse não apenas a formatação do documento, como também sua manipulação.

Uma vez contempladas essas características, a XML passa a fornecer um meio completo para a elaboração e distribuição de documentos por toda a *Web*, sendo independente de plataformas e de sistemas. O objetivo era transformar o conceito da HTML, fornecendo à XML recursos adicionais para a distribuição de documentos.

B.3.4. Visão Comparativa: HTML ou XML

Comparando a Figura 78, que exhibe a sintaxe HTML, com a Figura 79, que exhibe a sintaxe XML, notamos que as linguagens são bem parecidas, ambas possuem *tags* para marcar as informações, mas a diferença fundamental que torna o documento XML “inteligente”, é que sua estrutura demonstra claramente qual o significado das informações.

Figura 78. HTML ou XML – (exemplo da sintaxe HTML)

Figura 79. HTML ou XML – (exemplo da sintaxe XML)

Apesar dos dois documentos conterem basicamente as mesmas informações, no documento escrito em HTML um *software* não pode reconhecer o significado dos dados inseridos livremente no texto. Isso não acontece com o documento XML, tanto um *software* como as pessoas podem ler o conteúdo do documento e interpretar corretamente seus significados, e responder as seguintes perguntas:

- Qual o fabricante do veículo ? **HONDA**
- Qual é o modelo ? **CIVIC LX 1.6 16V (4 portas)**
- Qual o ano de fabricação ? **2000**
- Qual o ano do modelo ? **2000**
- Qual o tipo de câmbio utilizado ? **MECÂNICO**
- Qual a cor do carro ? **GRAFITE**
- Este veículo teve um único proprietário ? **SIM**
- Qual o preço de venda ? **R\$ 27.000,00**

Nota-se que as *tags* foram criadas na medida da necessidade do desenvolvedor do documento, eliminando a limitação imposta pela HTML, em que apenas as *tags* definidas pela linguagem podem ser usadas. Observamos também que na sintaxe XML, foram utilizadas *tags* auxiliares para tornar as características do modelo e preço do veículo melhor especificadas.

B.3.5. Estrutura

A estrutura do documento XML, consiste de duas partes:

1. DTD (*Document Type Definition*);
2. A definição dos elementos, que é o uso efetivo das “*tags*” declaradas pela DTD, conforme mostrado no exemplo do veículo, Figura 79.

A função da DTD é definir todas as *tags* que um documento XML pode conter, determinando a ordem em que elas devem aparecer e se são obrigatórias ou opcionais. O DTD também é responsável por armazenar diversos elementos utilizados nos documentos, como atributos que as *tags* podem conter e as entidades utilizadas nos documentos.

Com a DTD, é possível ao *software* processador da XML realizar a validação de um documento, verificando se ele foi criado corretamente, segundo os padrões estabelecidos. Quando a HTML é utilizada em uma página da *Web*, não existe a necessidade de atrelá-la a um DTD. Uma vez que suas *tags* são fixas, elas podem ser controladas internamente pelo próprio *browser*, de forma transparente ao desenvolvedor.

A XML, devido à característica de permitir a criação de *tags*, obriga a criação do DTD, para que o *browser* (ou qualquer outro processador XML) possa verificar a validade das *tags* utilizadas. O DTD possibilita ao *browser* trabalhar como um “compilador” de documentos XML, conforme mostrado na Figura 80.

Figura 80. *Browser* validando um documento XML

Com o auxílio da DTD, o *browser* verifica todos os detalhes, linha a linha, armazenados no documento XML, e ao encontrar um erro, ele será informado na tela.

A elaboração do DTD é o fator que possibilita a criação de diferentes linguagens de marcação por meio da XML.

É com a utilização dos DTDs que se pretende estabelecer padrões de marcação para as mais diversas áreas, possibilitando que todas as empresas de um mesmo ramo de atividade (ou até de outro) possam se comunicar de maneira unificada, criando vários tipos de linguagem de marcação a partir da XML.

A DTD pode ser declarada em um arquivo externo (.dtd) ou estar definida dentro do próprio documento XML (.xml). A DTD deve reconhecer os seguintes blocos de construção:

- ELEMENTOS: São blocos de construção utilizados tanto em HTML quanto em XML. Exemplos de elementos em HTML são “BODY” e “TABLE”. Em XML pode se referir a qualquer *tag* criada.
- TAGS: São os elementos marcadores, que possuem abertura e encerramento, como no exemplo: <VEICULO>, </VEICULO>.
- ATRIBUTOS: Fornecem informações extras sobre os elementos e são inseridos nas tags iniciais de um elemento. Possuem um nome e um conteúdo. Um exemplo de atributo pode ser .
- ENTIDADES: São variáveis usadas para definir textos comuns.

- NOTAÇÕES: Existem duas formas de notação: PCDATA que possibilita o armazenamento de texto que será interpretado pelo analisador do documento e CDATA que armazena texto não interpretado pelo analisador.

Na realidade, a DTD é a gramática dos documentos XML, outra maneira de especificar esta gramática excedendo inclusive as funcionalidades da DTD, é através dos esquemas XML (XML Schemas) [28].

B.3.6. Utilização e Importância

A XML não é apenas mais uma linguagem de marcação como a HTML, ela possibilita a utilização de vários recursos importantes. A possibilidade do desenvolvedor definir marcadores personalizados torna o documento “mais inteligente”, dando significado ao texto entre os marcadores. Enquanto a HTML apenas trata de especificar a formatação de uma palavra ou um trecho de texto, a XML trata de fornecer seu significado, e esse é o aspecto mais importante da XML.

Outro aspecto significativo é que ambas linguagens, HTML e XML, são padrões abertos. Nenhuma empresa pode monopolizá-las. Os documentos criados em XML pertencem ao seu criador. Sua função principal é criar condições para permitir uma padronização na descrição das informações. Uma vez padronizada a estrutura do documento, é possível com a utilização de linguagens de programação, interpretar e manipular o conteúdo do documento.

B.3.7. Padrões XML existentes no mercado

Conforme [25], já existem diversos padrões no mercado, criados a partir da XML, que visam estabelecer uma forma única entre os envolvidos na manipulação dos documentos, de maneira

que interfaces adequadas possam trocar dados automaticamente. Alguns padrões definidos atualmente, são:

- **MathML** – *Mathematical Markup Language* utilizada para descrição de fórmulas matemáticas. Disponível em: <<http://www.w3.org/TR/REC-MathML>>. Acesso em: 11 fev. 2005.
- **MML** – *Medical Markup Language* utilizada para troca de dados entre diferentes instituições envolvidas com clínica médica. Disponível em: <http://www.medxml.net/MML_E1028.pdf>. Acesso em: 11. Fev. 2005
- **GXL** – *Graph Exchange Language* utilizada para padronizar a troca de elementos gráficos, possibilitando interoperabilidade de diferentes ferramentas. Disponível em: <<http://www.gupro.de/GXL>>. Acesso em: 11 fev. 2005.
- **HL7** – *Health Level 7* utilizada para interoperabilidade entre sistemas de informação na área de cuidados médicos. Disponível em: <<http://www.hl7.org>>. Acesso em: 11 fev. 2005.
- **CML** – *Chemical Markup Language* utilizada na marcação de informações da área química. Disponível em: <<http://www.xml-cml.org>>. Acesso em: 11 fev. 2005.

Outras padrões existentes no mercado em diferentes áreas de interesse podem ser encontrados no portal XML. Disponível em: <<http://www.xml.org>>. Acesso em: 11 fev. 2005.

B.4. RDF e RDF Schema

B.4.1. Conceito

O objetivo da RDF (*Resource Description Framework*), desenvolvida pelo *World Wide Web Consortium* (W3C), é prover um modo comum para descrever informações que possam ser lidas e compreendidas por computadores através de aplicações.

A RDF define um modelo simples para descrever relacionamentos entre recursos em termos de nome das propriedades e valores. As propriedades RDF podem ser vistas como os atributos dos recursos.

O modelo de dados RDF não provê nenhum mecanismo para declarar estas propriedades, nem provê qualquer mecanismo para definir os relacionamentos entre estas propriedades e outros recursos. Este é o papel do *RDF Schema*.

A comunidade de Sistemas de Potência necessita descrever atributos de vários objetos, por exemplo: resistência, reatância e fluxo de MW. A declaração destas propriedades (atributos) e as suas semânticas correspondentes estão definidas no contexto do RDF como um *RDF Schema*. Um *schema* não somente descreve as propriedades do recurso, mas também pode definir o tipo dos recursos que estão sendo descritos (transformador, gerador, disjuntor, etc.).

Um *RDF Schema* foi definido pela codificação do modelo abstrato do CIM com o vocabulário do *RDF Schema*. Considerando-se que a RDF é genérica bastante para descrever conceitos UML, a conversão é direta [4], [29], [30].

B.4.2. Importação e Exportação das Informações de um Sistema de Potência

Com a formalização e padronização sobre o *CIM RDF Schema* (IEC 61970-501) [17], informações contidas num EMS podem ser convertidas para exportação, como um documento XML. Este documento é referenciado como um documento CIM/XML. Todas as etiquetas "tags" (descrições de recursos) usadas no documento CIM/XML são providas pelo esquema *CIM RDF*. O modelo de documento de troca CIM/XML resultante, pode ser analisado gramaticalmente e as informações importadas para um outro sistema [4].

Nos testes de interoperabilidade do CIM/XML, [8], [9], [10], [11], empresas participantes como a ALSTON ESCA e a Siemens, forneciam arquivos com dados dos sistemas de potência no formato CIM/XML, que eram intercambiados, armazenados e validados entre os sistemas SCADA/EMS envolvidos nos testes. A Tabela 18, exibe um fragmento representando um trecho de uma linha de transmissão.

Tabela 18. Representação de trecho de uma LT em CIM/XML

```
<cim:ACLineSegment rdf:ID="_8B0EEF4223822B47">
  <cim:Conductor.r>1.071225</cim:Conductor.r>
  <cim:Conductor.g0ch>9430.137022</cim:Conductor.g0ch>
  <cim:Conductor.x>11.188350</cim:Conductor.x>
  <cim:Conductor.x0>31.966714</cim:Conductor.x0>
  <cim:ConductingEquipment.BaseVoltage rdf:resource="#_B1F6DF75308C943A"/>
  <cim:Naming.name>1</cim:Naming.name>
  <cim:Conductor.bch>203.532755</cim:Conductor.bch>
  <cim:Conductor.r0>7.141500</cim:Conductor.r0>
  <cim:Conductor.b0ch>-42211.085861</cim:Conductor.b0ch>
  <cim:Conductor.gch>12013.289725</cim:Conductor.gch>
  <cim:ACLineSegment.MemberOf_Line rdf:resource="#_AD4F9632B2E4AC51"/>
</cim:ACLineSegment>
```

ANEXO C – TIPOS DE REATORES NUCLEARES

C.1. Reator de Ebulição de Água (BWR – *Boiling Water Reactor*)

Conforme [20], estes reatores foram originalmente projetados por *Allis-Chambers* e *General Electric* (GE). O projeto da GE sobreviveu, visto que todas as unidades da *Allis-Chambers* estão, no momento, com paradas (*shutdown*) programadas. A primeira planta comercial da GE nos Estados Unidos foi na baía de Humboldt (perto de Eureka) na Califórnia. Outros fornecedores de projetos de BWR no mundo são: *ASEA-Atom*, *Kraftwerk Union* e *Hitachi*. Os reatores comerciais, do tipo BWR, podem ser encontrados na Finlândia, na Alemanha, na Índia, no Japão, no México, nos Países Baixos, na Espanha, na Suécia, na Suíça e em Formosa. O reator, do tipo BWR, permite tipicamente ferver a maior parte da água no reator. A temperatura operacional do reator é de aproximadamente 570°F (299°C) produzindo vapor a uma pressão de aproximadamente 1.000 libras por polegada quadrada. Os reatores atuais (BWR) têm saídas elétricas de 570 a 1.300 MWe.

A Figura 81, apresenta um Sistema Reator de Ebulição de Água (*BWR System*).

Figura 81. Sistema Reator de Ebulição de Água (*BWR System*) - [20]

Na Figura 81, a água circula através do núcleo do reator (*Reactor Core*), absorvendo calor e sendo aquecida suficientemente, para se transformar em vapor. Os separadores de vapor na parte superior do reator, removem a água do vapor.

O vapor passa então através das linhas principais de vapor (*Main Steam Lines*) às Turbinas Geradoras (*Turbine Generators*). O vapor vai primeiramente a uma turbina de alta pressão (*HP*) menor, passa pelos separadores de umidade (não mostrados), passando então, pelas 3 turbinas de baixa pressão (*LP*), que são comuns numa planta de 1000 MWe. As turbinas estão conectadas ao gerador por um longo eixo ou biela (*shaft*).

O gerador produz a eletricidade, aproximadamente 20.000 Volts de corrente alternada. Esta potência elétrica é então direcionada a um transformador, que eleva a tensão para 230 kV ou

345 kV. Posteriormente a potência é remetida a uma subestação para distribuição da energia elétrica gerada.

O vapor, após a passagem através das turbinas, condensa-se então no condensador (*Condenser*), o qual está num vácuo que é resfriado pelo oceano, pelo mar, por um lago ou pela água de um rio. O vapor condensado é bombeado aos aquecedores de alimentação de água de baixa pressão (mostrados mas não identificados). A água passa então, às bombas de alimentação de água (*Feedwater Pumps*) que por sua vez, bombeiam a água ao reator, para reinício do ciclo.

O BWR é o único em que o controle das varetas ou hastes (*Control Rods*), utilizados na parada programada do reator, mantêm uma distribuição de potência uniforme através do reator, é introduzido por baixo através de um sistema de alta pressão operado hidraulicamente. O BWR tem também uma piscina de supressão (*Torus*), que é usada para remover o calor liberado se ocorrer um evento onde grandes quantidades de vapor sejam liberadas do reator, ou do sistema de circulação, utilizado para circular a água através do reator.

C.2. Reator de Água Pressurizada (PWR – *Pressurized Water Reactor*)

Conforme [21], estes reatores foram projetados originalmente pela *Westinghouse Bettis Atomic Power Laboratory* para aplicações militares, e posteriormente pela *Westinghouse Nuclear Power Division* para aplicações comerciais. A primeira planta comercial de PWR nos estados unidos foi *Shippingport* (Pennsylvania), que foi operada pela *Duquesne Light* até 1982.

Além da *Westinghouse*, a *Asea Brown Boveri-Combustion Engineering* (ABB-CE), a *Framatome*, a *Kraftwerk Union*, a *Siemens* e a *Mitsubishi* construíram este tipo de reator pelo mundo.

A Figura 82, apresenta um Sistema Reator de Água Pressurizada (*PWR System*).

Figura 82. Sistema Reator de Água Pressurizada (*PWR System*) - [21]

O reator de água pressurizada (PWR) tem 3 sistemas refrigeração separados e somente no sistema de refrigeração do reator, espera-se ter radioatividade.

O sistema de refrigeração do reator, mostrado dentro da estrutura de retenção (*Containment Structure*), consiste em 2, 3 ou 4 “circuitos de refrigeração” conectados ao reator, cada um contém uma bomba de refrigeração do reator, e o gerador de vapor. O reator aquece a água, que passa acima e fora do agrupamento de combustível, de uma temperatura de aproximadamente de 530°F (277°C) a uma temperatura perto de 590°F (310°C). A ebulição não é permitida ocorrer, exceção feita às bolhas menores chamadas de ebulição nuclear (*nucleate boiling*). A pressão é mantida, por um mecanismo de pressurização (não mostrado) conectado ao sistema de refrigeração do reator, em aproximadamente 2.250 libras por polegada quadrada através de um sistema de aquecimento e de pulverização no pressurizador. A água do reator é bombeada ao gerador de vapor e passa através de tubos. O sistema refrigeração do reator é o único que contém materiais radioativos. Tipicamente os PWR possuem 2, 3, ou 4 circuitos de refrigeração do reator dentro da estrutura de retenção.

No sistema refrigeração secundário (que inclui o sistema principal de vapor e os sistemas de alimentação de água e os condensadores), uma água gelada é bombeada do sistema de alimentação de água e passa sobre à parte externa dos tubos de geração de vapor, que é aquecida e convertida em vapor. O vapor passa então através de uma linha principal de vapor à turbina, que está conectada e movimenta o gerador. O vapor da turbina condensa-se em um condensador. A água condensada é bombeada através dos aquecedores de alimentação de água de baixa pressão, a seguir às bombas de alimentação de água, depois aos aquecedores de alimentação de água de alta pressão, e por último ao gerador de vapor. O diagrama da Figura 10, simplifica o processo mostrando somente o condensador, uma bomba, e o gerador de vapor.

O condensador é mantido em um vácuo usando bombas de vácuo ou ejetores de ar. A refrigeração do vapor é obtida pelo bombeando a água gelada do condensador através das bombas de circulação de água, que fazem a sucção da água fornecida do oceano, do mar, do lago, do rio ou da torre refrigeração (mostrada).

REFERÊNCIAS

- [1] MCMORRAN, A. W.; AULT, G. W.; ELDERS, I. M.; FOOTE, C. E. T.; BURT, G. M; MACDONALD, J. R. Translating CIM XML power system data to a proprietary format for system simulation. **IEEE Transactions on Power Systems**, v. 19, p. 229-235, feb. 2004.

- [2] SAXTON, T. L.; BECKER, D. **Common Information Model (CIM): CIM 10 Version**. Palo Alto: EPRI, 2001. 267 p. (Final report, 1001976).

Disponível em: <http://www.epri.com/attachments/283626_1001976.pdf>.
Acesso em: 14 jun. 2005.

- [3] CUNHA JR., C.A. S.; REIN JR., O.; JARDINI, J. A.; MAGRINI, L. C. Electrical Utilities Control Center Data Exchange with ICCP and CIM/XML. In: 2004 IEEE/PES TRANSMISSION AND DISTRIBUTION CONFERENCE AND EXPOSITION – LATIN AMERICA, 2004, São Paulo, Brasil. **Anais**. São Paulo: IEEE/PES T&D 2004 Latin America, 2004. 6 p.

Disponível em:
<http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=30876&arnumber=1432388&count=176&index=46>.
Acesso em: 23 jun. 2005.

- [4] DE VOS, A.; WIDERGREN, S. E.; ZHU, J. XML for CIM model exchange. In: IEEE POWER ENGINEERING SOCIETY INTERNATIONAL CONFERENCE ON POWER INDUSTRY COMPUTER APPLICATION, 22., 2001, Sidney, Australia. **Anais**. 2001. p. 30-37.

Disponível em: <<http://www.langdale.com.au/PICA/CIMXML.pdf>>.
Acesso em: 14 jun. 2005.

- [5] BERRY, T. Standards for Energy Management System Application Program Interfaces. In: INTERNATIONAL CONFERENCE ON ELECTRIC UTILITY DEREGULATION AND RESTRUCTURING AND POWER TECHNOLOGIES, 2000, Londres, Inglaterra. **Anais**. 2000. p. 159.

- [6] IEC 61970 – ENERGY MANAGEMENT SYSTEM APPLICATION PROGRAMMING INTERFACE (EMS-API) – Final Draft International Standard. Part 301: Common Information Model (CIM) Base, v 10.4, Nov. 2003.

Disponível em: <<http://cimuser.org>>.
Acesso em: 14 jun. 2005.

- [7] RAY, E. T. Aprendendo XML. Tradução Daniel Vieira. Rio de Janeiro: Campus, 2001. 372 p.

- [8] SAXTON, T. L.; BECKER, D. **Report on the Common Information Model (CIM) Extensible Markup Language (XML) Interoperability Test #4: The Power of the CIM to Exchange Power System Models.** Palo Alto: EPRI, 2002. 66 p. (Final Report, 1007351).

Disponível em: <<http://www.epriweb.com/public/000000000001007351.pdf>>.
Acesso em: 14 jun. 2005.
- [9] SAXTON, T. L.; BECKER, D. **Report on the Common Information Model (CIM) Extensible Markup Language (XML) Interoperability Test #1: The Power of the CIM to Exchange Power System Models.** Palo Alto: EPRI, 2001. 52 p. (Technical Progress, 1006161).

Disponível em: <<http://www.epriweb.com/public/000000000001006161.pdf>>.
Acesso em: 14 jun. 2005.
- [10] SAXTON, T. L.; BECKER, D. **Report on the Common Information Model (CIM) Extensible Markup Language (XML) Interoperability Test #2: The Power of the CIM to Exchange Power System Models.** Palo Alto: EPRI, 2001. 58 p. (Technical Progress, 1006216).

Disponível em: <<http://www.epriweb.com/public/000000000001006216.pdf>>.
Acesso em: 14 jun. 2005.
- [11] SAXTON, T. L.; BECKER, D. **Report on the Common Information Model (CIM) Extensible Markup Language (XML) Interoperability Test #3: The Power of the CIM to Exchange Power System Models.** Palo Alto: EPRI, 2001. 56 p. (Final Report, 1006217).

Disponível em: <<http://www.epriweb.com/public/000000000001006217.pdf>>.
Acesso em: 14 jun. 2005.
- [12] PODMORE, R.; HIRSCH, P. **CIM Data Views for Bulk Power Systems.** Palo Alto: EPRI, 1999. 70 p. (Final Report, TR-113384).
- [13] BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML Guia do Usuário.** Tradução Fábio Freitas da Silva. Rio de Janeiro: Editora Campus/Elsevier – 12^a. tiragem, 2000. 472 p.
- [14] KOSTIC, T.; FREI, C.; PREISS, O.; KEZUNOVIC, M. Scenarios for Data Exchange using Standards IEC 61970 and IEC 61850. In: UCA USERS GROUP MEETING, 2004, Paris, França. **Anais.** 2004. 5 p.

Disponível em: <<http://eppe.tamu.edu/k/cnf/UCA2004.doc>>.
Acesso em: 14 jun. 2005.
- [15] INTERNATIONAL ELECTROTECHNICAL COMMISSION. **Draft IEC 61850 Communication Networks and Systems in Substations Part 7-3: Basic communication structure for substations and feeder equipment – Common data classes.** IEC, 2001. 49 p.

- [16] HIRSCH, P.; PODMORE, R.; ROBINSON, M. **User Guide for CIM XML File Importer and Exporter Version 3.4.1**. Palo Alto: EPRI, 2003. 60 p. (EPRI Level 2 Report, 1009037).
- [17] LANGDALE. **CIM RDF Schema (Version 10 – RDF/HTML formats)**.
Disponível em: <<http://www.langdale.com.au/CIMXML/>>.
Acesso em: 14 jun. 2005.
- [18] PODMORE, R; ROBINSON, M. **Introduction to CIM**. PowerData Corporation and Incremental Systems Corporation, 2000. 94 p.
Disponível em: <http://www.epri.com/attachments/292151_CimIntroduction.pdf>.
Acesso em: 14 jun. 2005.
- [19] LANGDALE. **CIM Validate (Version 20010909a)**.
Disponível em: <<http://www.langdale.com.au/validate/>>.
Acesso em: 14 jun. 2005.
- [20] BOILING WATER REACTOR (BWR). Produced by Joseph Gonyeau.
Disponível em: <<http://www.nucleartourist.com/type/bwr.htm>>.
Acesso em: 14 jun. 2005.
- [21] PRESSURIZED WATER REACTOR (PWR). Produced by Joseph Gonyeau.
Disponível em: <<http://www.nucleartourist.com/type/pwr.htm>>.
Acesso em: 14 jun. 2005.
- [22] FOWLER, M.; SCOTT, K. **UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos**. Tradução Vera Pezerico e Christian Thomas. Porto Alegre: Editora Bookman – 2^a. edição, 2000. 169 p.
- [23] OMG UNIFIED MODELING LANGUAGE SPECIFICATION: UML – MARCH 2003 – VERSION 1.5 – FORMAL/03-03-01. Produced by Object Management Group.
Disponível em: <<http://www.omg.org/docs/formal/03-03-01.pdf>>.
Acesso em: 14 jun. 2005.
- [24] MAGRINI, L. C. **Integração de Subsistemas para compor a Automação da Distribuição de Energia Elétrica**. 1999. 275 f. Tese (Doutorado em Engenharia Elétrica) – Escola Politécnica, Universidade de São Paulo, São Paulo, 1999.
- [25] FURGERI, S. **Ensino Didático da Linguagem XML**. São Paulo: Editora Érica, 2001. 278 p.

- [26] W3C. **Extensible Markup Language (XML) 1.0 (Third Edition) – W3C Recommendation 04 February 2004.**

Disponível em: <<http://www.w3.org/TR/2004/REC-xml-20040204/>>.

Acesso em: 14 jun. 2005.

- [27] GOLDFARB, C. F.; PRESCOD, P. **The XML Handbook – 5th Edition.** Prentice Hall, 2003. 1268 p.

- [28] W3C. **XML Schema Part 0: Primer Second Edition – W3C Recommendation, 28 October 2004.**

Disponível em: <<http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/>>.

Acesso em: 14 jun. 2005.

- [29] W3C. **RDF Vocabulary Description Language 1.0: RDF Schema – W3C Recommendation 10 February 2004.**

Disponível em: <<http://www.w3.org/TR/2004/REC-rdf-schema-20040210/>>.

Acesso em: 14 jun. 2005.

- [30] W3C. **RDF Primer – W3C Recommendation 10 February 2004.**

Disponível em: <<http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>>.

Acesso em: 14 jun. 2005.