The CIM and Asset Modeling

The CIM provides a rich set of asset and network model-related classes which can be leveraged in organizing information related to the health of a wide variety of electric utility industry assets, from transformers and breakers to cables and tap changers. This summary explores the basic approach taken by the existing CIM in modeling assets and their participation in the electrical grid.

Organization of the CIM

The CIM is an electric utility industry standard semantic model, which covers information related to the entire realm of power system operation, from generation to customer, from the control center to the back office. It is expressed in Unified Modeling Language (UML) [9], [10], a language used for describing data structure and application structure, behavior and architecture as well as business processes. Three IEC standards comprise the CIM. The IEC 61970 standard describes the components of a power system at an electrical level and the relationships between each component. The IEC 61968 standard extends this model to cover other aspects of power system software data exchange such as asset management, work scheduling and customer billing. And the IEC 62325 standard extends both these models to cover the data exchanged between participants in electricity markets. These three standards, 61970, 61968 and 62325 are collectively known as the CIM.

The CIM data model is defined in a single, unified UML model, expressed in Enterprise Architect, where each standard has its own package. Three IEC Technical Committee 57 (TC57) working groups, Working Group 13 (WG13), Working Group 14 (WG14) and Working Group 16 (WG16), jointly maintain the UML and independently maintain the IEC Standard documents for which they are responsible. WG13 maintains 61970, WG14 has 61968 and WG16 is responsible for 62325. Coordination of the work of three groups is facilitated by a formalized model management process to ensure model integrity and by cross-representation on conference calls and at meetings which encourages a uniformity of modeling approach. Figure 1, a screen snapshot of the CIM UML, shows the high level organization of the CIM into standard-specific packages. This CIM UML reference is based on CIM UML version iec61970cim16v25_iec61968cim12v07a_iec62325cim03v01.

Figure 1

High-level CIM packages
CIM Support for Asset Health Information

Modeling of the full range of information applicable to asset health requires parts of both the 61970 and 61968 standards:

- The modeling of the electrical network and the behavior of its equipment is described in 61970-301 Common Information Model (CIM) Base [11].
- The modeling of assets and the information related to them is described in 61968-4 Interfaces for records and asset management [12], [13] and 61968-11 Common information model (CIM) extensions for distribution [14].

The 61970 standard and its associated UML packages (Domain, Core, Wires and Meas) provide the framework for organizing real-time flow and state information related to the asset as it is deployed in the field, allowing insight into the field conditions to which the asset has been exposed. The 61968 standards, and their associated packages in the CIM UML model (Asset, AssetInfo and Common), supply the framework for organizing information from an asset perspective and include classes for the asset itself, for manufacturer and model information, for inspection and test records and for lifecycle activities. Combined they provide the basis for defining the structure and organization of a robust model for asset health-related information.

Figure 2 is a screen snapshot of the sub-packages of the 61970 and 61968 packages of the CIM. The main 61968 and 61970 packages are highlighted in blue and under them the packages containing classes of interest for asset health modeling are highlighted in green.
Figure 2
CIM packages supporting asset health modeling
The content of each of the asset health-related packages is overviewed below:

- **The 61970 Domain package** contains definitions of datatypes that are used in describing the attributes of classes in all other packages of the CIM. Its CIM description is:

  The domain package define primitive datatypes that are used by classes in other packages. Stereotypes are used to describe the datatypes. The following stereotypes are defined:

  - <<enumeration>> A list of permissible constant values.
  - <<Primitive>> The most basic data types used to compose all other data types.
  - <<CIMDatatype>> A datatype that contains a value attribute, an optional unit of measure and a unit multiplier. The unit and multiplier may be specified as a static variable initialized to the allowed value.
  - <<Compound>> A composite of Primitive, enumeration, CIMDatatype or other Compound classes, as long as the Compound classes do not recurse.

- **The 61970 Core package** contains basic building block classes. The `IdentifiedObject` class, which is the parent of all other CIM classes that have identifiers, is in the Core package as are the parent classes for equipment, connectivity and containment. The three naming-related classes that support the ability for different applications to have different names for the same object are also a part of the Core package. Its CIM description is:

  Contains the core PowerSystemResource and ConductingEquipment entities shared by all applications plus common collections of those entities. Not all applications require all the Core entities. This package does not depend on any other package except the Domain package, but most of the other packages have associations and generalizations that depend on it.

- **The 61970 Wires package** is home to the classes that describe the electrical characteristics for many different types of equipment. The classes of the Wires package are typically children of the `Equipment` class in the Core package. The CIM description of the Wires package is:

  Contains entities that describe dynamic measurement data exchanged between applications.

- **The 61968 Common package** includes a set of classes, some of them parent classes with more specific children classes in other packages, which provide a basic set of building blocks for modeling data exchanged within a utility enterprise. Classes in the Common package include those that describe location, activity, organization, person, schedule, ownership. The Common package description in the CIM is:

  This package contains the information classes that support distribution management in general.

- **The 61968 Assets package** has the foundational classes for the description of the physical characteristics of tangible electric utility resources. It contains the generic classes used for describing assets (`Asset` and `AssetContainer`), as well as classes that support asset-related actions and the entities that might perform them. The Asset package description in the CIM is:

  This package contains the core information classes that support asset management applications that deal with the physical and lifecycle aspects of various network resources (as opposed to power system resource models defined in IEC61970::Wires package, which support network applications).

- **The 61968 AssetInfo package** contains classes that allow the definition of asset-specific or asset type-specific characteristics, like nameplate information. The AssetInfo package description in the CIM is:

  This package is an extension of Assets package and contains the core information classes that support asset management and different network and work planning applications with specialized AssetInfo
subclasses. They hold attributes that can be referenced by not only Assets or AssetModels but also by ConductingEquipments.

**CIM Asset Health Classes**

Using the classes of those packages, a full-featured model supporting asset health information can be built. At the center of the model is the asset itself, which is described by using two classes in the Assets package, `Asset` and `AssetContainer`, which allow the identification of an asset and its component parts. The CIM description of those two classes is:

- **Asset**
  
  Tangible resource of the utility, including power system equipment, various end devices, cabinets, buildings, etc. For electrical network equipment, the role of the asset is defined through PowerSystemResource and its subclasses, defined mainly in the Wires model (refer to IEC61970-301 and model package IEC61970::Wires). Asset description places emphasis on the physical characteristics of the equipment fulfilling that role.

- **AssetContainer**

  Asset that is aggregation of other assets such as conductors, transformers, switchgear, land, fences, buildings, equipment, vehicles, etc.

Figure 3 shows the CIM UML model of those two classes on the left along with an instance diagram showing conceptually how they would support the instance representation of an asset with two components. Note in the UML that `Asset Container` both inherits from `Asset` and has a relationship with `Asset`, allowing the repetitive nesting of component assets within component assets. The `Asset` and `AssetContainer` classes are intended to represent the simple fact that a physical asset exists and to allow the description of a few types of information which any asset might have: identifying numbers, geographic location, type, condition, status. The `Asset` class does not contain any information that relates to specific types or models of assets. The `Asset` and `AssetContainer` classes are the foundational classes from which an asset model is built. All other classes supplying asset-related information have associations to the asset model.
Information on asset characteristics or asset model characteristics is provided by the AssetInfo class, or more accurately by the asset type-specific child classes of AssetInfo which are in the AssetInfo package. A collection of classes (AssetModel, ProductAssetModel and Manufacturer) are used to specify the model and manufacturer of an asset. The CIM definitions of those classes are:

- **AssetInfo**
  
  Set of attributes of an asset, representing typical datasheet information of a physical device that can be instantiated and shared in different data exchange contexts:
  - as attributes of an asset instance (installed or in stock)
  - as attributes of an asset model (product by a manufacturer)
  - as attributes of a type asset (generic type of an asset as used in designs/extension planning).

- **AssetModel**
  
  Model of an asset, either a product of a specific manufacturer or a generic asset model or material item. Datasheet characteristics are available through the associated AssetInfo subclass and can be shared with asset or power system resource instances.

- **ProductAssetModel**

  Asset model by a specific manufacturer.

- **Manufacturer**

  Organisation that manufactures asset products.

Figure Error! No text of specified style in document.4 shows the UML model of the participating classes and expands the asset conceptual instance representation to include asset and asset model characteristics as well as model and manufacturer information. The information modeled by the AssetInfo child classes is intended to be specific to the type of asset (switch, cable, transformer, meter, etc.) and is intended to be used in three different ways: to describe the characteristics of a specific asset, to describe the characteristics of a specific asset model or to describe the characteristics of a generalized asset used in the field construction design and planning process. Note the relationship of both AssetModel and Asset to AssetInfo. The AssetModel relationship reflects the use of AssetInfo for describing asset model-specific information, the Asset relationship leverages AssetInfo to describe information specific to a given asset.

The classes shown in Figure Error! No text of specified style in document.4 are the essential classes for describing the physical asset itself. Two other types of classes, those that support the modeling of the results of asset-related procedures and those that provide information about the environment in which an asset has been operated are needed to fully describe the organizational structure of an asset’s condition-related information.
Basic asset plus model and asset-specific characteristics – UML and instance representation

Figure Error! No text of specified style in document.4
There are a multitude of asset-related activities whose results have bearing on understanding asset health: factory, field and lab tests, field inspections and various types of maintenance tasks. In the CIM, activities related to assets are described using the Procedure class. The results of activities related to assets are described using the ProcedureDataSet class or its children. Both the Procedure class and the ProcedureDataSet class are contained in the Asset package. The CIM definitions of those classes are:

- **Procedure**
  
  Documented procedure for various types of work or work tasks on assets.

- **ProcedureDataSet**
  
  A data set recorded each time a procedure is executed. Observed results are captured in associated measurement values and/or values for properties relevant to the type of procedure performed.

Figure 5 shows the UML of these activities- and results-related classes and illustrates where they would fit in the conceptual instance representation.
Figure 5
Asset plus asset-related activities and results – UML and instance representation
All the CIM classes used so far in building the asset health model have come from the IEC 61968 standard. Gaining insight into the environment in which an asset is or has been operating requires the use of IEC 61970 classes, which represent the network model in terms of network components, their behavior and their connectivity. IEC 61970 also includes support for measurement points representing flows, voltages, states, alarms or other values which can be associated to either distinct locations in the network or to specific devices. The Equipment class and the ConductingEquipment class, both of which are child classes of PowerSystemResource, are three of the fundamental IEC 61970 classes which form the basis of the model needed for asset health. These classes, which are contained in the Core package, and their equipment-specific child classes, most of which are in the Wires package, facilitate the identification of network components and their behavior. Two other classes fundamental to asset environment and usage modeling are the Measurement class which supports identifying the existence of a measuring point and its network position and the MeasurementValue class which represents an actual value at a given measuring point. Both the Measurement class and the MeasurementValue class are contained in the Meas package. The CIM definitions of these five classes are:

- **Equipment**
  The parts of a power system that are physical devices, electronic or mechanical

- **ConductingEquipment**
  The parts of the AC power system that are designed to carry current or that are conductively connected through terminals.

- **PowerSystemResource**
  A power system resource can be an item of equipment such as a switch, an equipment container containing many individual items of equipment such as a substation, or an organisational entity such as sub-control area. Power system resources can have measurements associated.

- **Measurement**
  A Measurement represents any measured, calculated or non-measured non-calculated quantity. Any piece of equipment may contain Measurements, e.g. a substation may have temperature measurements and door open indications, a transformer may have oil temperature and tank pressure measurements, a bay may contain a number of power flow measurements and a Breaker may contain a switch status measurement.
  
  The PSR - Measurement association is intended to capture this use of Measurement and is included in the naming hierarchy based on EquipmentContainer. The naming hierarchy typically has Measurements as leaves, e.g. Substation-VoltageLevel-Bay-Switch-Measurement.

  Some Measurements represent quantities related to a particular sensor location in the network, e.g. a voltage transformer (PT) at a busbar or a current transformer (CT) at the bar between a breaker and an isolator. The sensing position is not captured in the PSR - Measurement association. Instead it is captured by the Measurement - Terminal association that is used to define the sensing location in the network topology. The location is defined by the connection of the Terminal to ConductingEquipment.

  If both a Terminal and PSR are associated, and the PSR is of type ConductingEquipment, the associated Terminal should belong to that ConductingEquipment instance.

  When the sensor location is needed both Measurement-PSR and Measurement-Terminal are used. The Measurement-Terminal association is never used alone.

- **MeasurementValue**
  The current state for a measurement. A state value is an instance of a measurement from a specific source. Measurements can be associated with many state values, each representing a different source for the measurement.
The UML with the fundamental asset operating environment classes added at the left is shown in Figure 6. The corresponding conceptual instance representation augmented with operating environment information is shown in Figure 7. The connection of an asset to its operating environment is represented by the bold line between Conducting Equipment and Asset Identification in the instance representation of Figure 6. In the UML of Figure 6, this relationship is the bold lined association between the PowerSystemResource class and the Asset class. This relationship is the key relationship that allows information about the environment in which an asset “lives” or has “lived” and the manner in which it has been used to be correlated with an asset’s physical characteristics as input into asset health analytics.
A number of additional relationships also appeared in the UML of Figure Error! No text of specified style in document.6:

- Between PowerSystemResource and AssetInfo
- Between Measurement and Asset
- Between Measurement and Procedure
- Between Measurement and ProcedureDataSet

The relationship between PowerSystemResource and AssetInfo is intended to support the population of 61970 equipment behavior parameters from 61968 typical asset characteristics. The relationship between Measurement and Asset recognizes the need to express operating environment conditions as having direct bearing on assets. The Measurement to Procedure and MeasurementValue to ProcedureDataSet relationships are a case of CIM class re-use: the set of Measurement classes provides a good way of describing types of values (analsogs and discrete) and supporting multiple instances of them. Certain types of asset activity results may benefit from leveraging this approach to modeling.
**Asset Identity and Nameplate Information**

In creating templates, the focus is on developing a solid basic asset model useful for asset health. Four sample Transmission breakers will be used to help expose modeling requirements that can’t necessarily be discerned from a “class” view.

The most basic group of exchanged information is that of asset identity and nameplate:

1. Identity of the asset and each of its components
2. Nameplate information
   - asset type and model
   - component type and model
   - physical characteristics
   - date of manufacture

The CIM provides basic modeling support for both types of data listed above. The Asset and AssetContainer classes provide the asset and asset component identity information required by 1. and the AssetInfo class (and its children), the AssetModel, ProductAssetModel and Manufacturer classes together provide a model to support the information for 2. The details of CIM support for each type of data are explored below.

**Asset and Component Identity**

The Asset and AssetContainer classes provide very flexible building blocks for defining assets and their components. They also provide the ability to identify the type of asset and to describe essential asset-specific information that applies to assets of all types. The UML for the current CIM model is shown in Figure Error! No text of specified style in document.8. The attributes of interest are:

- .type, which allows the identification of the kind of asset. The .type attribute currently has a datatype of String
- .serialNumber, which also has a datatype of String
CIM Asset and AssetComponent classes

The CIM classes definitely support the ability to exchange the needed asset and asset component identity information, but they provide no guidance as to how instances might be constructed to represent any given type of asset. Since one of the purposes of the CIM is to promote application interoperability, guidance on construction of generic asset component models for common types of assets would be of significant value to the industry.

As an illustration of what this guidance might look like, component models are proposed for the sample Transmission breakers. Since an asset component model is a direct reflection of the physical and functional characteristics of the asset, the proposed breaker component models vary from fairly simple to quite complex depending on the type of breaker. To aid in understanding each proposed component model, a physical description of the type of breaker it represents precedes its component diagram.

SF6 Dead Tank Breaker

A dead tank circuit breaker, according to ANSI C37.100[15] is “A switching device in which a vessel(s) at ground potential surrounds and contains the interrupter(s) and the insulating mediums.” SF6 dead tank circuit breakers started being manufactured in the mid 1960s [16] and are in common use in the Transmission system today. Figure 9 shows a picture of an SF6 dead tank breaker.
The component model for the SF6 breaker, like those of all the breakers, reflects its physical construction, with a single AssetContainer instance representing the whole breaker which contains three AssetContainer instances, each representing a tank assembly. Each tank assembly AssetContainer instance has two Asset instances associated with it, one for each bushing. The instance model for the sample SF6 breaker is shown in Figure 10.

The value of the whole breaker Asset.type attribute is sF6DeadTankBreaker, which is indicative of a design decision that was made relative to asset (or more specifically, breaker) typing. The .type attribute of the Asset class affords the opportunity to provide meaningful asset categorization information. Agreement on the possible values of .type and their meanings is essential to any deployed integration solution and the definition of standard (or generic) types and their meanings is nearly as essential for successful definition of a CIM model for asset health. It is the .type attribute that

- allows applications to understand the structure of the component model of an asset
- allows validation of the appropriateness of the association that is made between asset and conducting equipment
- allows procedures and their results to be correctly associated to assets and asset components

Since the asset domain being used for instance samples is Transmission breakers, it would be useful to suggest a list of possible standard .types for use in that domain:

- sF6DeadTankBreaker
- sF6LiveTankBreaker
- bulkOilBreaker
- minimumOilBreaker
- airBlastBreaker
- vacuumBreaker

The specification of .type allows all applications in the integration environment to accurately anticipate what the component model of a given asset will look like. Using the SF6 dead tank breaker as an example, applications could expect that an AssetContainer instance with .type = sF6DeadTankBreaker would have three related AssetContainer instances with .type = sF6DeadTankBreakerTankAssembly, each of which have two related Asset instances with .type = bushing. It is also expected that the three tank assembly AssetContainer instances will have .positionNumber attribute values = 1, 2 and 3 and that the six bushing Asset instances will have .positionNumber attribute values = 1, 2, 3, 4, 5 and 6 which reflect the position of a given bushing with respect to the other bushings associated with the breaker.

![SF6 Dead Tank Breaker Component Model](image)

**Figure 10**
SF6 dead tank breaker component instance model

In constructing component models for Transmission breakers, a universal need for position numbering of components became apparent. The tank assemblies of breakers are identified by pole numbers. Bushings are numbered and power factor test results are reported using those numbers. It makes sense to believe that the need for position numbering is not limited to breakers. Any device with components connected to different phases probably needs such numbering. So the addition of a .positionNumber attribute to the Asset class is suggested.
The UML for the Asset class with the addition of .type enumerations for the types of breakers listed above (and their components) and the new .positionNumber attribute is shown in Figure Error! No text of specified style in document.11 below.

Figure Error! No text of specified style in document.11
CIM Asset class with proposed enumerated .type and new .positionNumber attribute

Bulk Oil Breaker

Bulk oil breakers are another type of dead tank circuit breaker, one where oil is used both as the interrupting medium and the insulating medium. Bulk oil breakers were among the earliest types of breakers used in Transmission applications and continued to be manufactured though the mid-1980s [17]. While decreasing in numbers, bulk oil breakers are still in common use in the Transmission system today. Figure Error! No text of specified style in document.12 shows a picture of a bulk oil breaker.
The component model for bulk oil breakers is more complex than that of an SF6 dead tank breaker. Since component modeling is a direct reflection of the physical and functional parts of the breaker, the fact that bulk oil breakers can be comprised of either a single three-pole tank or three single-pole tanks means there are two alternate models. The rules governing the assembly of a bulk oil breaker component model are:

- The whole breaker AssetContainer instance for the bulk oil breaker has a .type attribute value = bulkOilBreaker.
- Related to the whole breaker AssetContainer instance are either one or three AssetContainer instances.
  - If there is a single three-pole tank, the single AssetContainer instance has a .type attribute value = bulkOilBreakerThreePoleTankAssembly and a .positionNumber attribute value = 1. There are six Asset instances associated with the tank assembly AssetContainer instance, with .type attribute value = bushing, one for each bushing. Bushing Asset instances have .positionNumber attribute values = 1, 2, 3, 4, 5 and 6, which reflect the position of a given bushing with respect to the other bushings associated with the breaker.
  - If there are three single-pole tanks, there are three AssetContainer instances, with .type attribute value = bulkOilBreakerSinglePoleTankAssembly, each representing a tank assembly. Then, similar to the SF6 dead tank breaker model, the tank assembly AssetContainer instances have .positionNumber values = 1, 2 and 3 and each tank assembly AssetContainer instance has two Asset instances associated with it, with .type attribute value = bushing, one for each bushing. Bushing Asset instances have .positionNumber attribute values = 1, 2, 3, 4, 5 and 6,
which reflect the position of a given bushing with respect to the other bushings associated with the breaker.

The instance model for the sample bulk oil breaker is shown in Figure 13. The sample breaker has one tank assembly for each pole, so there are three tank assembly AssetContainer instances related to the whole breaker AssetContainer instance.
Figure 13
Bulk oil breaker component instance model
Air Blast Breaker

An air blast circuit breaker is a type of live tank breaker, which according to ANSI C37.100 [15] is “A switching device in which the vessel(s) housing the interrupter(s) is at a potential above ground.” Air blast breakers were primarily manufactured between the 1940s and early 1990s and at one time were the only type of circuit breaker suitable for use at voltages of over 345 kV. A significant portion of the installed population of air blast breakers has been retired, though a number of utilities still have them in use. Figure Error! No text of specified style in document.14 shows pictures of two styles of air blast circuit breakers.

Figure Error! No text of specified style in document.14
Air blast breakers
The modeling of air blast breaker components is more complex than that of SF6 dead tank breakers or even bulk oil breakers. The variable number of air reservoirs, insulating stacks and interrupting chambers among different models of air blast breakers means the component models can have more permutations. The rules governing the assembly of an air blast breaker component model are:

- The whole breaker AssetContainer instance for the air blast breaker has a .type attribute value = airBlastBreaker.
- Related to the whole breaker AssetContainer instance are either one or three AssetContainer instances.
  - If there is a single three-pole air reservoir, the single AssetContainer instance has a .type attribute value = airBlastBreakerThreePoleReservoir and a .positionNumber attribute value = 1. The air reservoir AssetContainer instance has three AssetContainer instances associated with it, with .type attribute value = airBlastBreakerInsulatingStackAssembly, one for each pole’s insulating stack assembly. The insulating stack assembly AssetContainer instances have .positionNumber values = 1, 2 and 3. Each insulating stack assembly AssetContainer has associated to it one or two Asset instances with .type attribute value = airBlastBreakerInterruptingAssembly, one for each interrupting assembly. Interrupting assembly Asset instances have .positionNumber attribute values starting at 1 and going up to 6 as needed, which reflect the position of a given interrupter with respect to the other interrupters associated with the breaker.
  - If there are three single-pole reservoirs, there are three AssetContainer instances, with .type attribute value = airBlastBreakerSinglePoleReservoir. The air reservoir AssetContainer instances have .positionNumber values = 1, 2 and 3. Each air reservoir AssetContainer instance has one to four AssetContainer instances associated with it, with .type attribute value = airBlastBreakerInsulatingStackAssembly, one for each insulating stack associated with the reservoir. Insulating stack AssetContainer instances have .positionNumber attribute values starting at 1 and going up to 4 as needed, with numbering reflecting the position of the insulating stack with respect to the other insulating stacks associated with its air reservoir. Each insulating stack AssetContainer has one or two Asset instances with .type attribute value = airBlastBreakerInterruptingAssembly, one for each interrupting assembly. Interrupting assembly Asset instances have .positionNumber attribute values starting at 1 and going up to 8 as needed, which reflect the position of a given interrupter with respect to the other interrupters associated with its air reservoir.

The instance model for a three-pole air blast breaker, the simplest variation of the air blast breaker which matches the top picture of Figure Error! No text of specified style in document.14, is shown in Figure 15. This sample breaker has one air reservoir for all three poles so there is only one air reservoir AssetContainer instance related to the whole breaker AssetContainer instance.
Figure 15
Air blast breaker (three-pole version) component instance model
The instance model for a more complicated air blast breaker, one with three single pole reservoirs and 8 breaks per phase (nearly matching the breaker in the lower picture in Figure Error! No text of specified style in document.14, which has 6 breaks per phase) is shown in Figure Error! No text of specified style in document.16. This sample breaker has three air reservoirs, one for each pole. Each air reservoir hosts four insulating stacks and each insulating stack has two interrupters.

*Figure Error! No text of specified style in document.16
Air blast breaker (three single-pole version) component instance model*
To Model or Not To Model...

The flexibility afforded by the CIM recursive modeling of assets and asset components creates an extremely flexible framework, a generic illustration of which is shown Figure 17 below. To leverage the flexibility of the CIM for real-world usefulness, standard asset component models need to be defined using the framework. The challenge becomes making appropriate decisions regarding what components of an asset merit modeling and what level of detail will best support interoperability.

![Diagram of CIM modeling structure](image)

**Figure 17**
Asset component flexibility provided by CIM

In creating standard asset component models, the desire is to provide definition at the optimal level of detail: one that is just detailed enough to support the accurate association of information of importance to the part of the asset to which it pertains, but not more detailed than that. The following “rule of thumb” questions were used in the development of the sample Transmission breaker component models and may be generally useful in determining what components of an asset merit individual modeling and what components should be viewed as simply a part of a larger assembly:

- Can a component be replaced independently of the assembly to which it belongs? If so, is there need for information about how the new component differs from the old component?
- Is there any testing or inspection done that relates specifically to the component and not to the assembly to which the component belongs?
- Can a component be put “in service” (connected to the grid) independently of other components?
- Does the any characteristic of the component (material, manufacturer, model, in-service date) matter in analyzing asset health?

The sample Transmission breaker component models developed using these guidelines appear to be reasonably accurate within the Transmission domain, but a great deal more work is required to bring knowledge to bear concerning the full range of utility breakers before it would be appropriate to view these breaker component models as suitable for standardization.
It should be noted as well that the quality of the asset component models will grow over time as they are actually used in the real world to support the organization of procedure results and the population of equipment characteristics in network models.

**Nameplate Information**

Nameplate information in the CIM is provided by the *AssetInfo* class and its children together with the *AssetModel*, *ProductAssetModel* and *Manufacturer* classes. As was mentioned in Section 3, the information modeled by the *AssetInfo* child classes is intended to be specific to the type of asset (switch, cable, transformer, meter, etc.) and is intended to be used in three different ways:

- to describe the characteristics of a specific asset
- to describe the characteristics of a specific asset model or
- to describe the characteristics of a generalized asset used in the field construction design and planning process.

Since the *AssetInfo* child classes are type-of-asset-specific and since the samples we are using to explore model suitability are breakers, the UML shown in Figure Error! No text of specified style in document.18 is what the CIM current has for switches (or breakers). In the case of breakers, the classes which are actually instantiated to support nameplate information are: *SwitchInfo*, *ProductAssetModel* and *Manufacturer*. 
SwitchInfo as a child of AssetInfo, can be used in an asset-specific way, a model-specific way or in a typical asset-specific way. Since the sample Transmission breakers are real physical devices, the use of the SwitchInfo class in their modeling will not be typical asset-specific. An attribute by attribute evaluation of the information identified as needed to support the asset health use cases is required to determine if SwitchInfo for the sample breakers should be asset-specific or model-specific or if both may be needed.

The specific attributes cited in the use case are:
- Type
- Serial number
- Model number
- Physical characteristics
- Manufacture date
- Mechanism type and model number
- Rated fault current

In addition to these attributes, a utility asset expert suggested that the following pieces of nameplate information could be valuable in breaker health evaluations:
- Rated maximum volts
- Rated continuous current
- Impulse withstand voltage (or BIL)
- Rated frequency
- Interrupting time
- Low pressure alarm level
- Lock-out pressure alarm level
- Mechanism type
- SF6 gas weight (for SF6 dead tank breakers)
- Volume of oil per tank (for bulk oil breakers)

The CIM itself suggests a couple additional attributes:
- Whether the breaker is single phase or three phase
- Whether the breaker is gang operated or not

The first two of the attributes listed above (type and serial number) are covered by the asset component model. The third item (model number) is modeled by the .modelNumber attribute of the ProductAssetModel class. The fourth bullet refers in general to “physical characteristics” which are assumed to be well covered by attributes farther down in the list. The fifth bullet refers to information (manufacture date) which will be addressed in the discussion on asset lifecycle state.

The list of items that remain to be evaluated to determine whether they are model or asset related are shown in Table 1. In discussions with the utility expert it was determined that all the attributes are model-specific. In other words, the attributes would have the same value across every breaker of a given model.

Table 1
Nameplate attributes and what they apply to

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Specific to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanism type and model number</td>
<td>model</td>
</tr>
<tr>
<td><strong>Rated fault current</strong></td>
<td>model</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>Rated maximum volts</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Rated continuous current</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Impulse withstand voltage (or BIL)</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Rated frequency</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Interrupting time</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Low pressure alarm level</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Lock-out pressure alarm level</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Mechanism type</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>SF6 gas weight (for SF6 dead tank breakers)</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Volume of oil per tank (for bulk oil breakers)</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Whether the breaker is single or three phase</strong></td>
<td>model</td>
</tr>
<tr>
<td><strong>Whether the breaker is gang operated or not</strong></td>
<td>model</td>
</tr>
</tbody>
</table>

This situation may not be true for assets of types other than breakers – the data needs to be carefully evaluated but it certainly simplifies the instance creation for breakers.

In considering the three possible purposes of the AssetInfo class and its children and the fact that a given asset might have both model- and asset-specific instances of an AssetInfo class related to it, it seemed wise to add model support for the distinction. A proposed .context enumerated attribute, with possible values of instance, model and planned, for the AssetInfo class is suggested to fulfill that purpose. While the sample breakers had only model-specific nameplate information, assets that have both model- and asset-specific nameplate information will benefit from the ability to distinguish the AssetInfo child instance related to the model from the one related to the asset itself. This will reduce the likelihood that a model-related instance is inadvertently and inappropriately updated because it is mistaken for an asset-related instance.

The existing SwitchInfo class CIM model does not have all the attributes listed above. It is suggested to add the following attributes to SwitchInfo:

- `.ratedImpulseWithstandVoltage`
- `.ratedFrequency`
- `.ratedInterruptingTime`
- `.lowPressureAlarm`
- `.lowPressureLockOut`
- `.mechanismType`
- `.oilVolumePerTank`
- `.gasWeightPerTank`

Complete nameplate information for both SF6 dead tank breakers and bulk oil breakers could be considered to include information related to their bushings. There is no existing normative CIM class that covers bushing information modeling, so it is suggested to add a BushingInfo class with the following attribute:
The UML model with the additions suggested above to support more complete breaker nameplate modeling for asset health is shown in Figure 19.

In Figure 19, and all the other figures where UML additions or modifications are proposed, the changes are indicated in one of two ways: either by line/text color (for classes and associations) or by the use of the <<Asset Health>> stereotype (for attributes and associations). Red is used to indicate a suggested addition, grey is used to indicate a suggested removal. The proposed changes indicated in Figure 19 are the ones mentioned above plus the change of multiplicity from 0..1 to 0..* on the AssetInfo end of the Asset to AssetInfo association. This change is suggested to support the possibility that some assets may have both asset-specific and model-specific nameplate information.

Hypothetical nameplates, created from the synthesis of real-world breaker nameplates, are shown in the Figures below for each of the four sample breakers. Each hypothetical nameplate is followed by the portion of the breaker’s instance model which supports the nameplate information.
Figure 19
CIM breaker nameplate model with proposed additional SwitchInfo attributes, new AssetInfo.context attribute and new BushingInfo class

SF6 Dead Tank Breaker

The hypothetical nameplate for the sample SF6 dead tank breaker is shown in Figure 20 with the corresponding CIM classes and attributes indicated by callouts. Information that is model-specific is shown in blue and asset-specific information is in green.
Figure 20
SF6 dead tank breaker nameplate
The instance model for the sample SF6 dead tank breaker is shown in Figure 21, where model-related classes are highlighted in blue. Note the conversion of the following values from English units to SI:

- psig to kPa (kilo Pascal) for pressures
- lb to kg for weights (which assumes the assets are on the surface of the earth so the conversion of weight to mass yields valid results)

**Bulk Oil Breaker**

The hypothetical nameplate for the sample bulk oil breaker is shown in Figure Error! No text of specified style in document.22 along with an indication of the corresponding CIM classes and attributes.

The instance model for the sample bulk oil breaker is shown in Figure Error! No text of specified style in document.23 where model-related classes are highlighted in blue. Note that for this breaker, the bushings have a model number.

**Air Blast Breaker**

The hypothetical nameplate for the sample three-pole air blast breaker is shown in Figure 24 and its CIM instance model in Figure 25.
Figure 21
SF6 dead tank breaker instance model for nameplate information
### General Electric Bulk Oil Circuit Breaker

<table>
<thead>
<tr>
<th>Type:      FK-121-20000-2</th>
<th>Rated Max. Volts: 121 kV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Serial No: 0202A9486-801</td>
<td>Rated Short Circuit Current: 20,000 A</td>
</tr>
<tr>
<td>Impulse Withstand (BIL): 550 kV</td>
<td>Rated Continuous Current: 1200 A</td>
</tr>
<tr>
<td>Voltage Range Factor (K): 1</td>
<td>Interrupting Time: 3 Cycles</td>
</tr>
<tr>
<td>Rated Frequency: 60 Hz</td>
<td>Operating Air Pressure</td>
</tr>
<tr>
<td></td>
<td>Normal: 140 +/- 3 psig Minimum: 90 +/- 3 psig</td>
</tr>
<tr>
<td>Compr. Governor Switch</td>
<td>Close Press: 125 +/- 3 psig Open: 140 +/- 3 psig</td>
</tr>
<tr>
<td>Low Pressure Alarm Switch</td>
<td>Close Press: 100 +/- 4 psig Open: 111 +/- 4 psig</td>
</tr>
<tr>
<td>Low Pressure Lock-Out Switch</td>
<td>Close Press: 100 +/- 5 psig Open: 90 +/- 3 psig</td>
</tr>
<tr>
<td>Height to Remove Bushing: 17 ft</td>
<td>Oil per Tank: 580 gal</td>
</tr>
<tr>
<td>Weight of Bushing: 375 lb</td>
<td>Weight of Breaker with Oil: 25,200 lb</td>
</tr>
<tr>
<td>Pneumatic Operating Mechanism Type: MA-13-6A</td>
<td></td>
</tr>
<tr>
<td>Clos. Control Valve Coil: 366A7011G10</td>
<td>90 to 130 VDC 5.4 A</td>
</tr>
<tr>
<td>Potential Trip Coil: (2) 132A1165P67</td>
<td>70 to 140 VDC 12.0 A</td>
</tr>
<tr>
<td>Wiring Diagram: 0172C6960P12</td>
<td>Manufacture Date: 01/01/1966</td>
</tr>
</tbody>
</table>

**Figure Error! No text of specified style in document.22**

Bulk oil breaker nameplate
Bulk oil breaker instance model for nameplate information
**Figure 24**

*Air blast breaker (three-pole version) nameplate*
The hypothetical nameplate for the sample air blast breaker with three single-pole reservoirs is shown in Figure 26 and its CIM instance model in Figure 27.
Figure 26
Air blast breaker (three single-pole version) nameplate
Other Nameplate Considerations

The modeling of medium is being proposed as an addition to the core breaker asset model. It appears that it would make sense to have medium explicitly modeled in cases either where it could potentially be the subject of tests or other procedures or where knowing the extent of the medium would be helpful. The *Medium* class currently exists in the CIM in the 61968 informative package called InfAssets. It is suggested to make the *Medium* class and *MediumKind* enumeration class normative and to add the following enumerations to *MediumKind*:

- oil
- sF6
- air

Medium was modeled for all the sample breakers. The core sF6 dead tank breaker asset component model, along with its medium modeling highlighted in grey, is shown in Figure 28. Note that in this breaker, the sF6 for each tank assembly is present both in the tank and in the bushing, but is not shared between tank assemblies. Had the sF6 been shared across tank assemblies, as it is in some sF6 breakers, there would have been only one *Medium* instance and it would have had a relationship with all three tank assembly *AssetContainer* instances and all six bushing *Asset* instances.
Figure 28
SF6 dead tank breaker instance model showing medium

Figure 29, Figure 30 and Figure 31, show the medium modeling for the sample bulk oil breaker, the sample three-pole air blast breaker and the three single-pole air blast breaker respectively.

Figure 29
Bulk oil breaker instance model showing medium
Figure 30
Air blast breaker (three-pole version) instance model showing medium
Figure 31
Air blast breaker (three single-pole version) instance model showing medium
Figure 32, Figure 33, Figure 34 and Figure 35 illustrate the full breaker asset component model plus sample lifecycle events for the four types of sample breakers.
Figure 32
SF6 dead tank breaker full basic CIM instance model
Figure 33
Bulk oil breaker full basic CIM instance model
Figure 34
Air blast breaker (three-pole version) full basic CIM instance model
Figure 35
Air blast breaker (three single-pole version) full basic CIM instance model

NOTE
The above content is background material used in EPRI reports on asset health from 2011 – 2014, contributed to the CIMug Asset Health Focus Community for the furtherance of the CIM standard by Pat Brown, EPRI