Lessons Learned Part 2:
Business Vocabulary Management

Comparison of IEC CIM and NRECA MultiSpeak


Scott Neumann
Partner, UISOL
Chairman, USNC IEC TC57


MultiSpeak Background

- MultiSpeak effort funded by National Rural Electric Cooperative Association (NRECA)
- NRECA has almost 1000 member electric cooperatives in 47 states
- Goal is to achieve out of the box integration for applications used by NRECA members
- MultiSpeak version 1.1 focus on back-office applications, and defined batch file data transfers
- MultiSpeak version 2.2 supports both batch and real-time transfers and adds operations-support apps
- MultiSpeak has official compliance testing and certification

IEC CIM Background

- Common Information Model development started through efforts of EPRI, vendors and utilities through EPRI CCAPI Task Force
- CIM was adopted by International Electrotechnical Commission working groups for use in standards development
- CIM has become an international standard, IEC 61970-301 through the efforts of TC57 WG13
- CIM being extended for Distribution and Markets by TC57 WGs 14 and 16
- CIM has been harmonized with IEC 61850 through the efforts of AHWG07

Comparing CIM and MultiSpeak

- Where CIM covers transmission, generation and distribution, MultiSpeak is distribution focused
- **Best point of comparison is between IEC 61968 (WG14) and MultiSpeak, where the common focus is information exchanges related to distribution systems**
- MultiSpeak is focused to meet the needs of electric cooperatives in the US, while IEC 61968 is focused towards all utilities, including IOUs and the international marketplace
- IEC 61968 is transport independent while MultiSpeak is transport specific, this has trade-offs will respect to implementation utilization of EAI infrastructures and compliance testing
- IEC 61968 leverages the OAG namespace for ERP integration

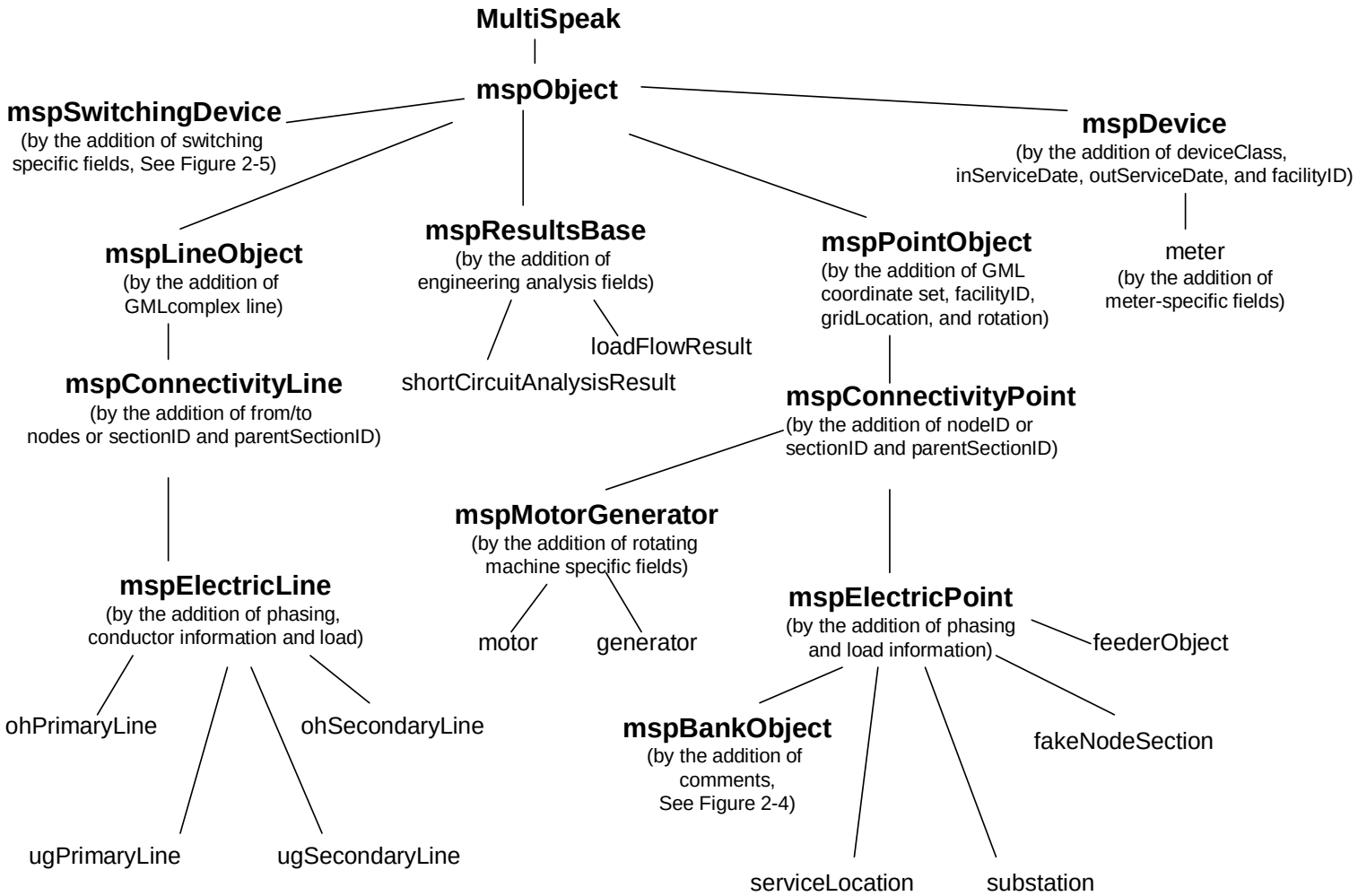
Common to MultiSpeak and IEC 61968

- Both focus on interfaces between applications, as opposed to data structures internal to applications
- Supporting models which define classes, simple types and complex types
- Use of XML Schema for definition of messages
- Messages have a control area and a payload
- Use of nouns and verbs for definition of messages (although actual nouns and verbs are different between the two)
- Leverage GML as defined by the Open GIS Consortium

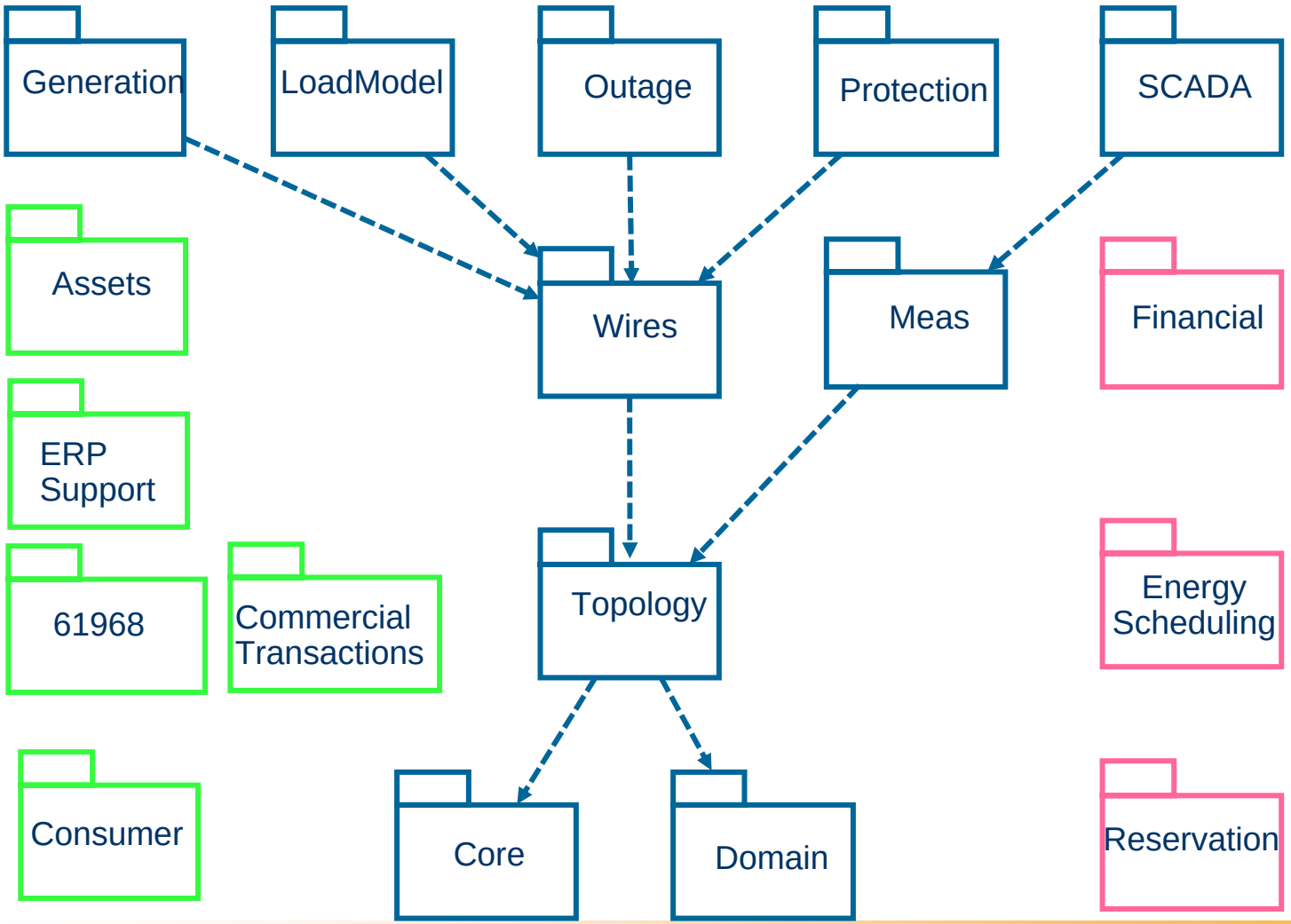
Key Model Differences

	MultiSpeak	CIM
Model Management	Model managed using XML Schema with XML Spy	Model managed with Rational Rose, where RDFS and XML Schemas are generated
Object identification	objectId found in mspObject class	Naming class used to manage names for virtually all CIM classes
Class Hierarchy	mspObject is parent class for mspSwitchingDevice, mspLineObject, mspPointObject, mspDevice and mspResultsBase, where each 'msp' class may have subclasses.	Organized using packages. Naming is parent for PowerSystemResource, whose descendant classes include Equipment and ConductingEuipment
Relationships	Inheritance relationships and 'Lists' and 'Banks' for aggregations	Wide variety of associations and aggregations are defined and managed in model
Connectivity	Supports both section-oriented (section, parent section) and node-oriented (from – to) connectivity	ConductingEquipment have terminals which are grouped into ConnectivityNodes – does not support section oriented topology
Asset Modeling	Asset related attributed included within class definitions as needed or through simple grouping (e.g. SwitchBank)	Asset model implemented where PowerSystemResources can be comprised of one or more Asset instances
Graphical Modeling	mspLineObject has GMLcomplex line and mspPointObject has grid location and rotation	Managed as an attribute of an instance, may have multiple representations

MultiSpeak Class Hierarchy

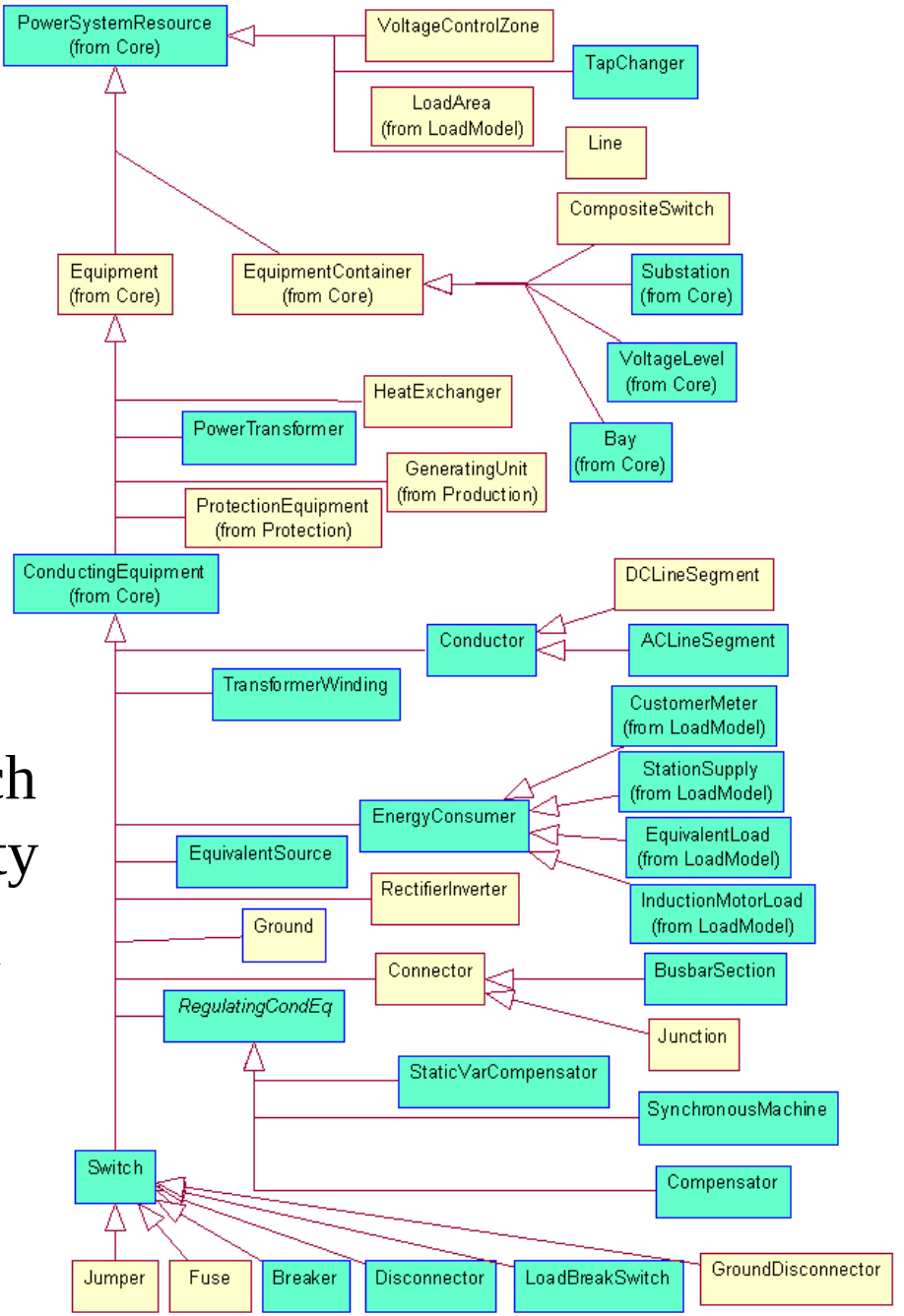


CIM Packages



CIM: Wires

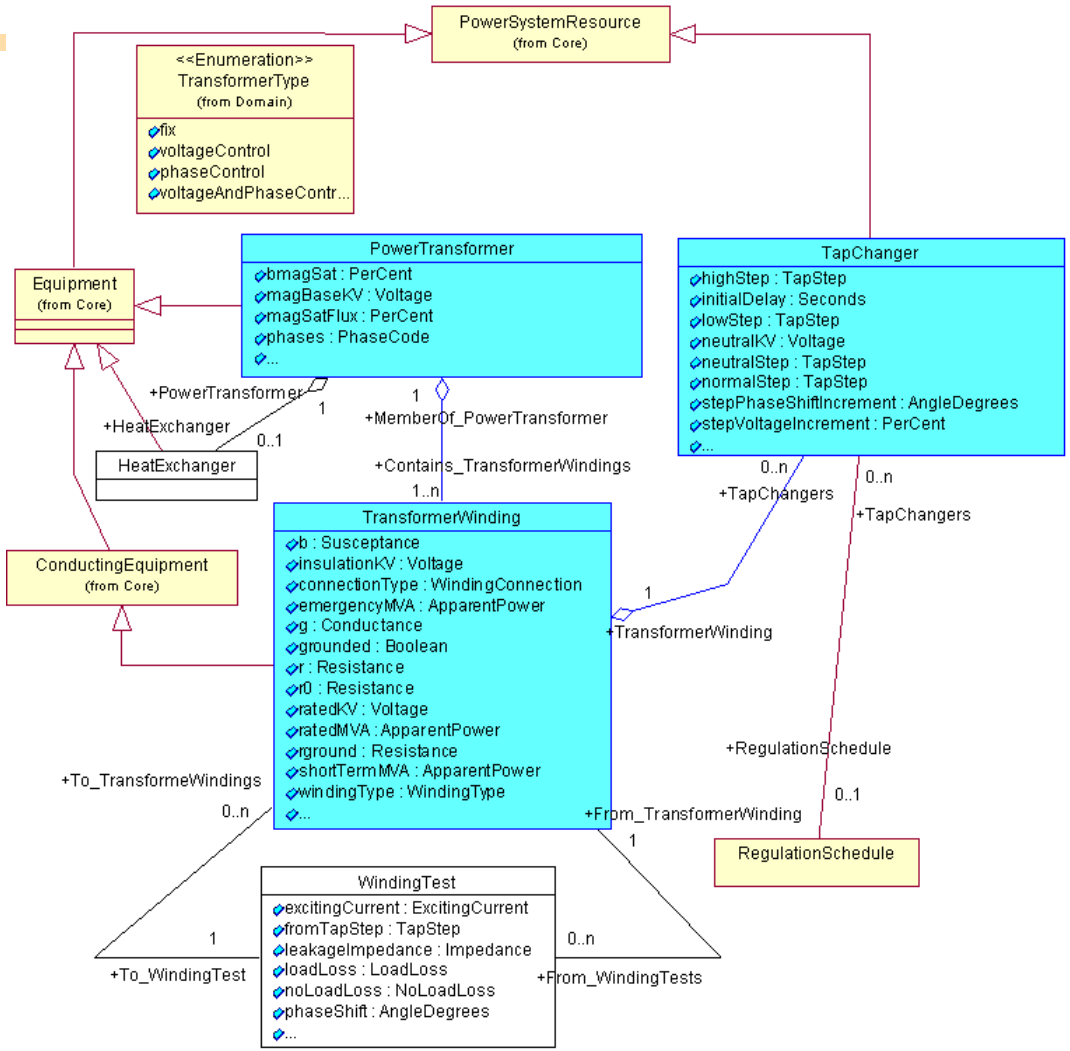
- Class hierarchy from CIM Wires package
- PowerSystemResource is key super class for objects which have electrical roles
- ConductingEquipment is super class for objects which define electrical connectivity
- Associations can be seen in other views of model



UISOL™

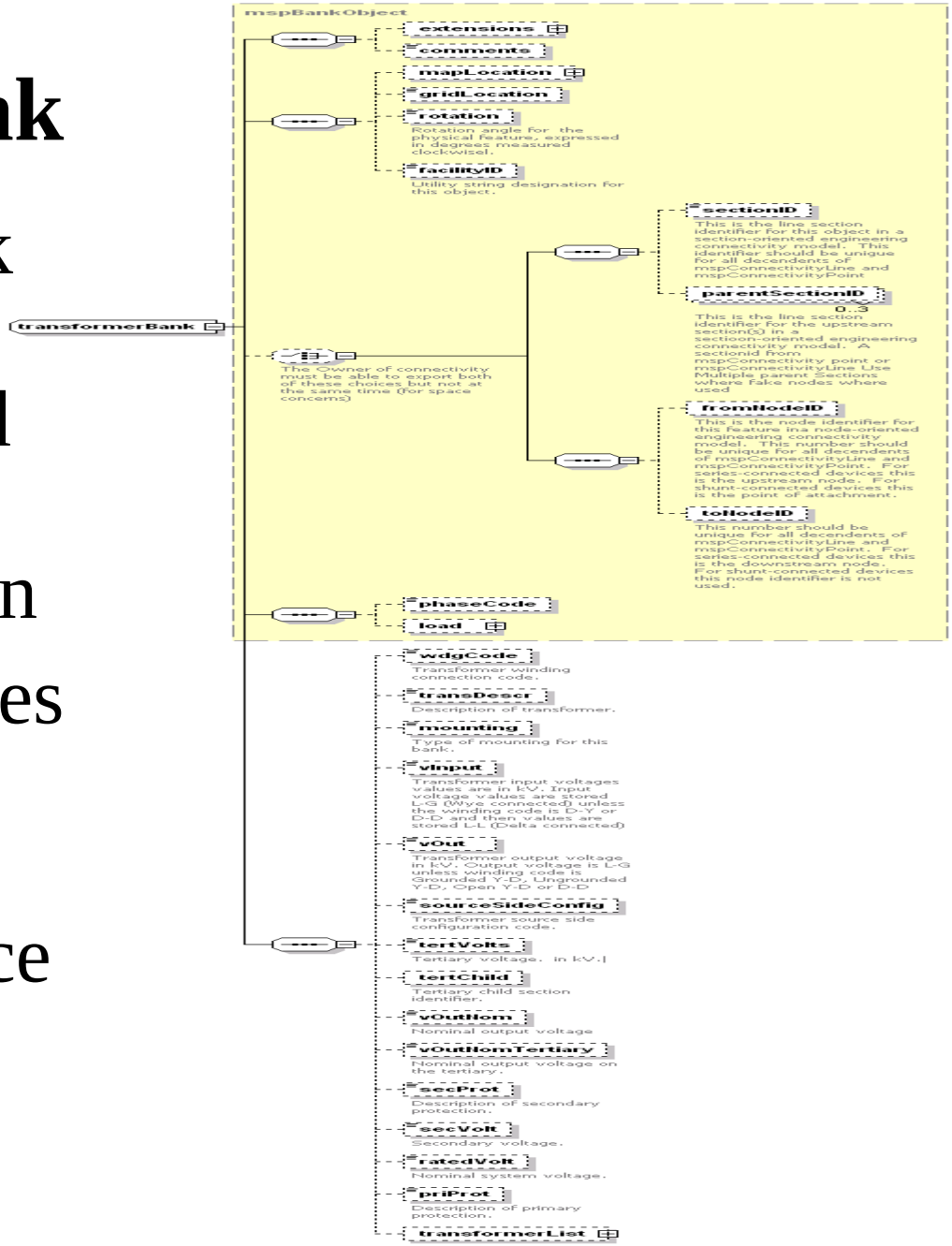
CIM: Transformers

- Transformer model from CIM Wires package
- Inheritance and associations are maintained within model
- Similar diagrams are also maintained for LineModels, Equipment Containment and Regulating Equipment



MultiSpeak: transformerBank

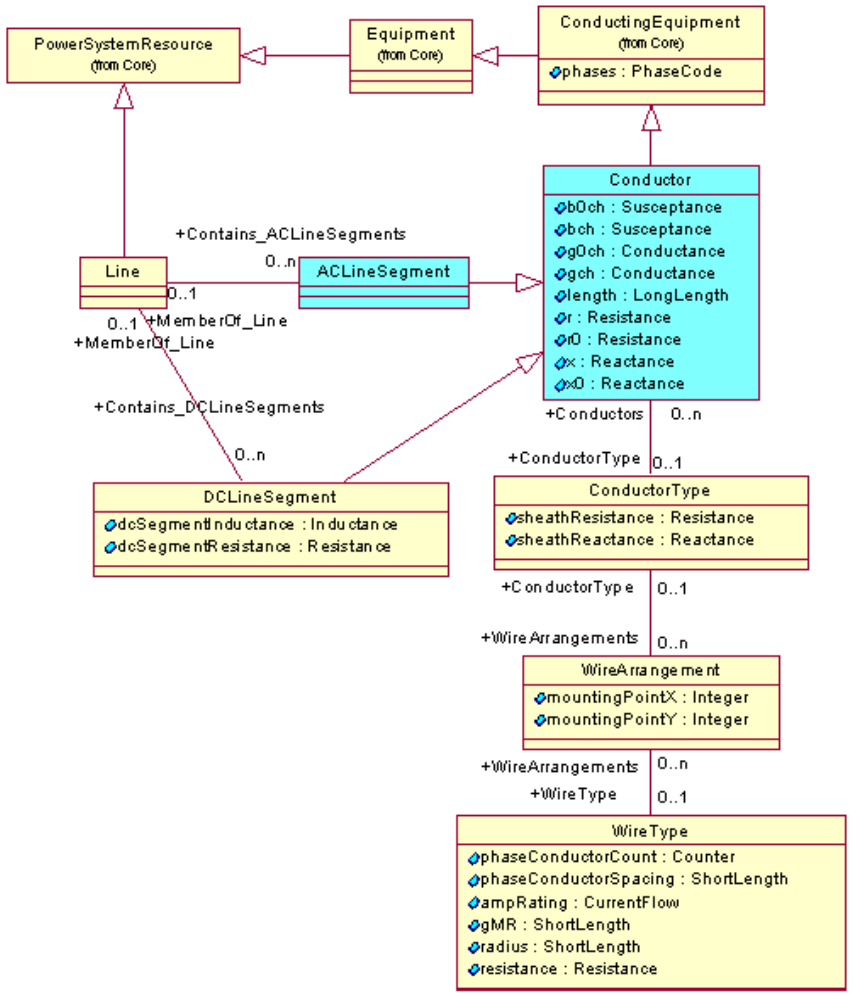
- transformerBank has a list of transformers and defines voltages and configuration
- tranformer defines asset related attributes and simple impedence model



UISOL™

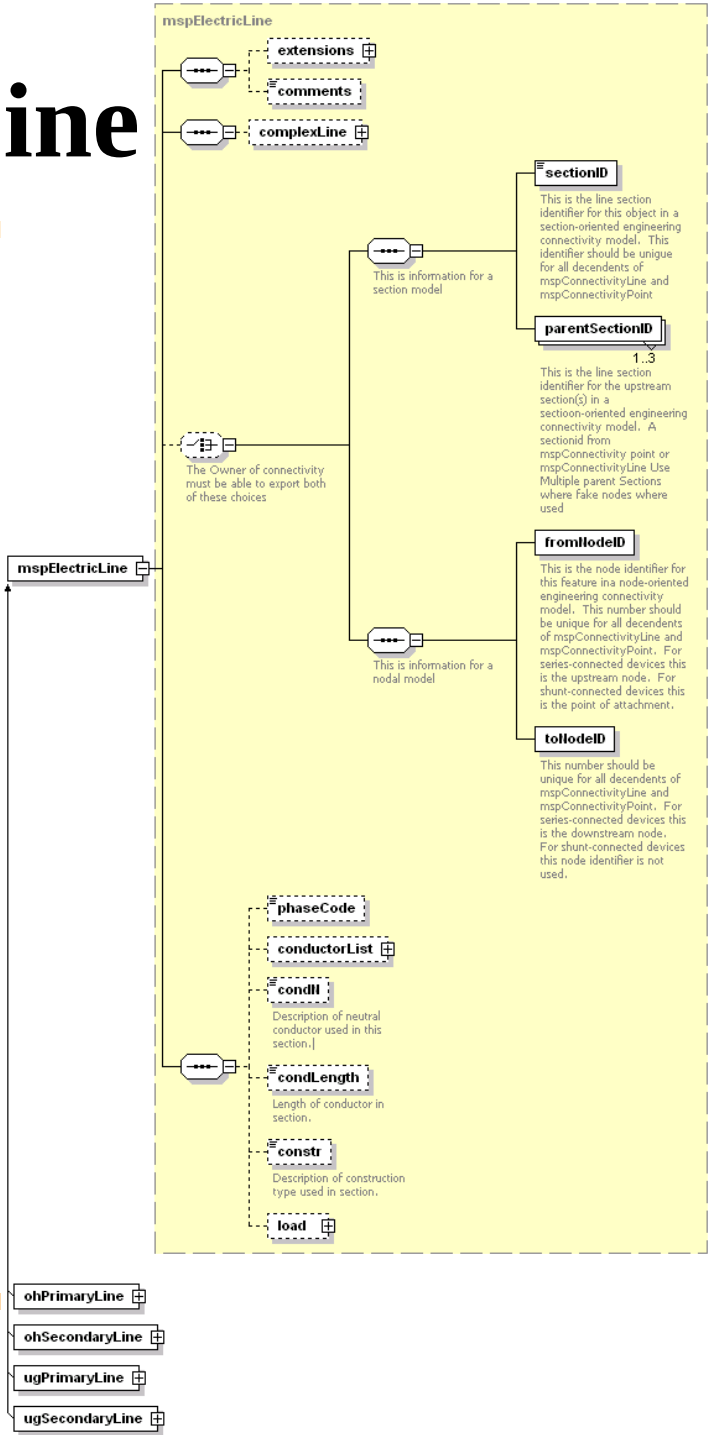
CIM: LineModel

- Line comprised of a set of ACLineSegments or DCLineSegments
- ACLineSegments and DCLineSegments are Conductors with a defined impedence and terminals for connectivity
- Conductor has ConductorType with WireArrangements and WireTypes
- May have multiple geometries (geographic and schematic)



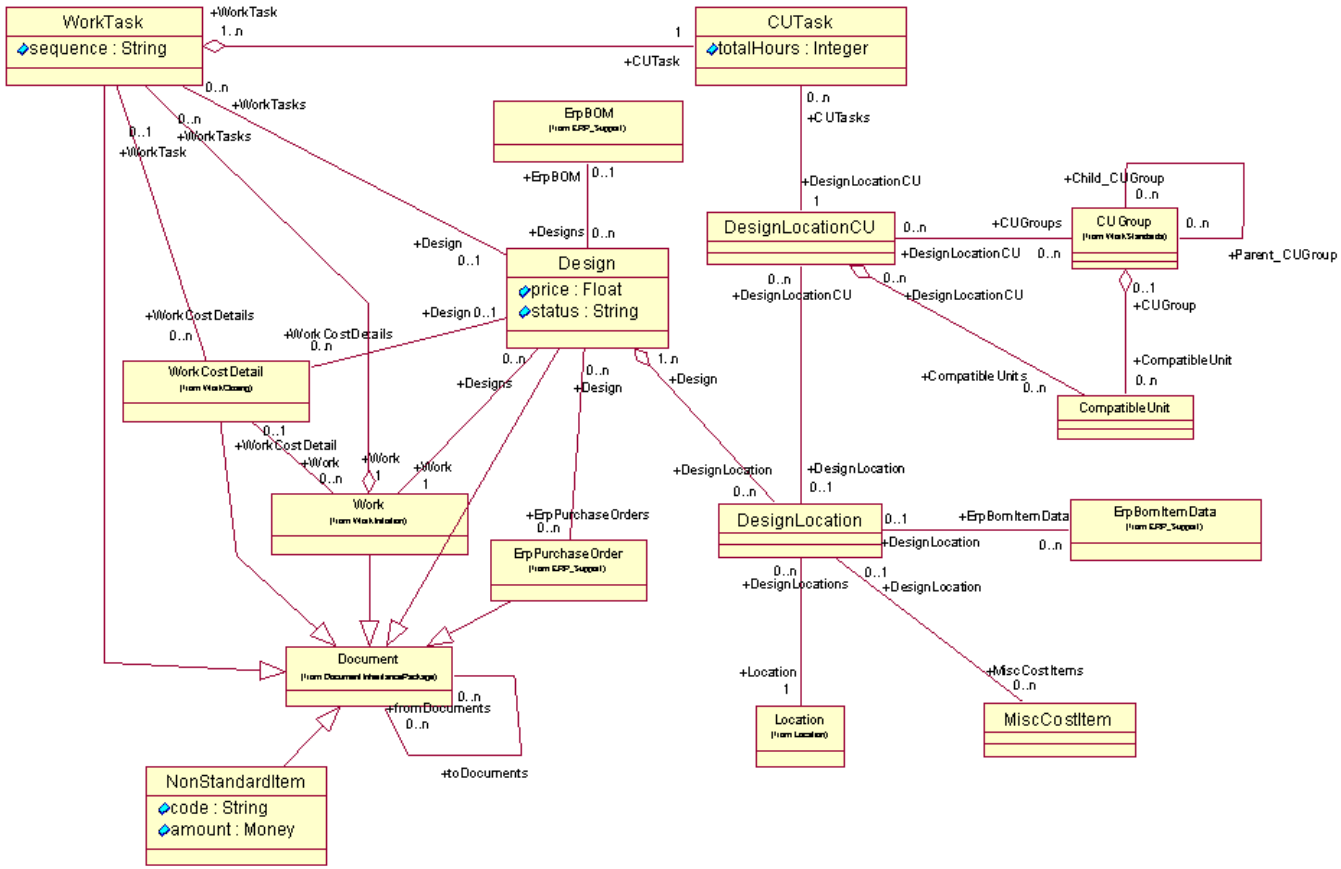
MultiSpeak: ElectricLine

- Inherits from mspLineObject, which defines geometry
- ElectricLine as four subclasses
- ElectricLine has a list of Conductors, with a Conductor for each phase
- Impedence indirectly managed by conductorType, construction, length
- Connectivity is defined by segment and by node



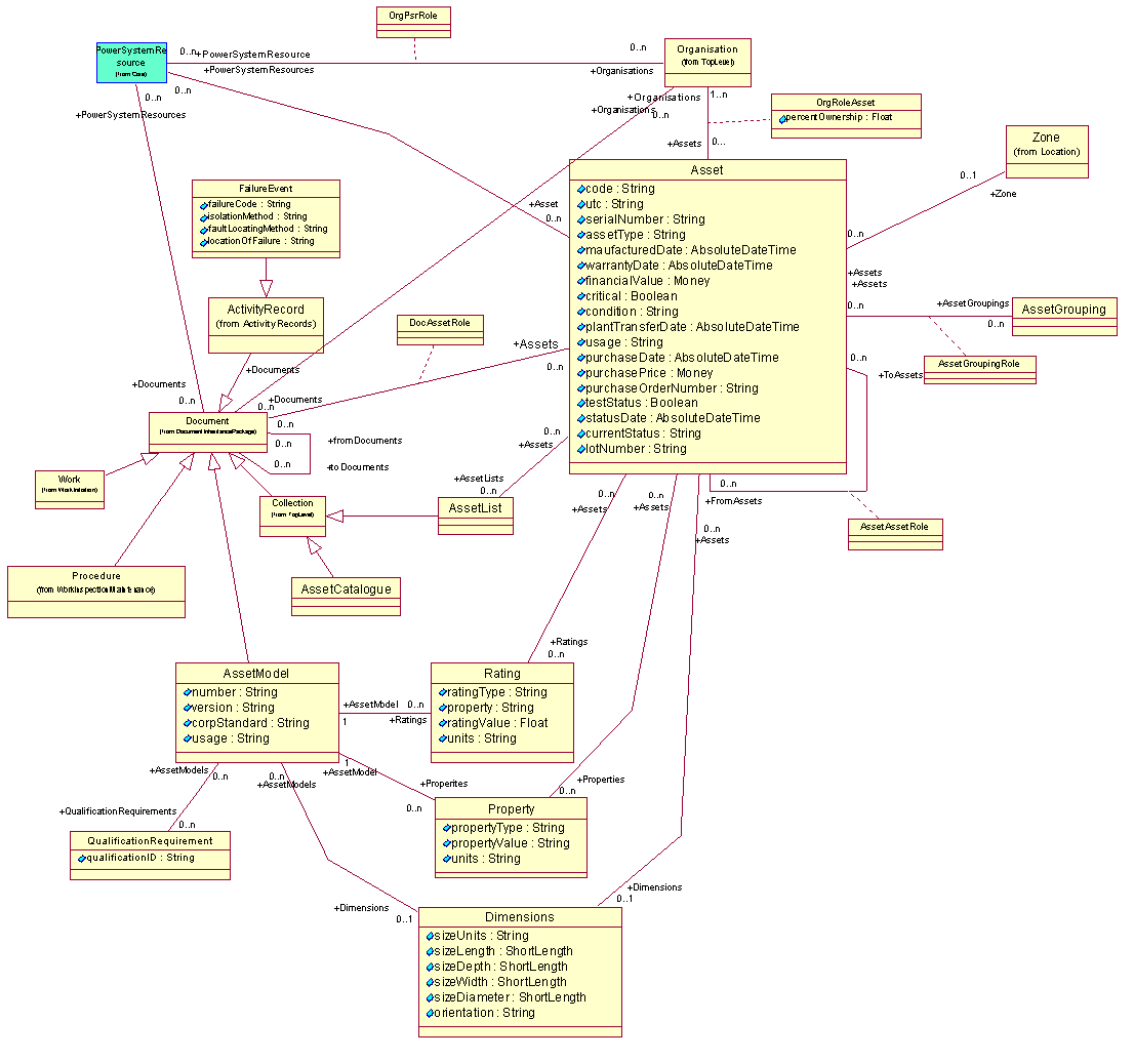
CIM: Work

- 61968 has rigorous modeling of work, with six sub packages for initiation, scheduling, **design,** closing, service and standards



CIM: Assets

- 61968 has rigorous asset model
- Asset model important to work management
- Power system resources are comprised of one or more assets



UISOL™

More Model Comparisons …

MultiSpeak	CIM	Key differences
switchDeviceBank, switch	**Switch, Asset**	CIM Switch with child Assets is roughly equivalent to a MultiSpeak switchDeviceBank comprised of a set of switch objects. Both have subclasses for fuses and breakers.
mspElectricLine	**Conductor**	MultiSpeak has ohPrimaryLine, ohSecondaryLine, ugPrimaryLine, ugSecondaryLine, where each has a list of Conductors CIM Conductor has subclasses ACLineSegment and DCLineSegment, with r, r0, x, x0, bch, b0ch, gch, g0ch Note that use of Conductor is very different in these models.
transformer, transformerBank	**PowerTransformer, TransformerWinding, WindingTest, Asset**	CIM impedence model includes b, g, r, r0, x, x0 MultiSpeak identifies priVoltsLo, priVoltsHi,secVoltsLo, secVoltsHigh, pcb
customer	**Customer**	MutliSpeak model currently more complete for AMR integration
meter	**CustomerMeter**	MutliSpeak model currently more complete for AMR integration
workOrder	**Work**	Where MultiSpeak workOrder is focused on staking, 61968 has a much broader treatment of distribution work management
outageEvent	**OutageReport**	OutageReport supports multiple steps and associatd tracking of customers affected.
customerCall	**TroubleTicket**	TroubleTicket tracks information needed for callbacks

Example: Breaker Attributes

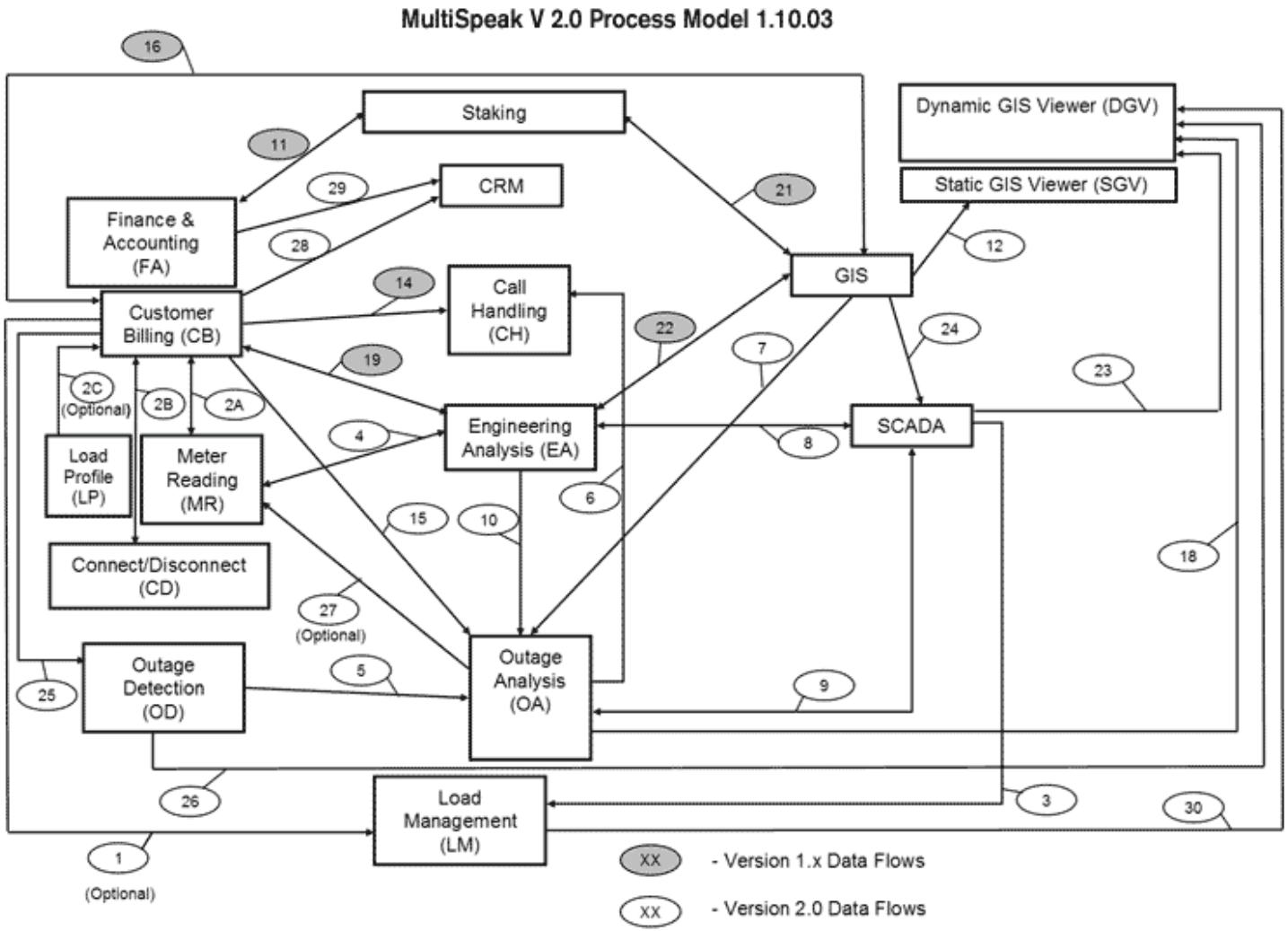
MultiSpeak (breaker):

- objectID
- maxContAmp
- phase
- position
- comments
- eqEquipment
- facilityID
- ratedVolt
- operVolt
- manufacturer
- mounting
- bypassExists
- lastService

CIM (Breaker):

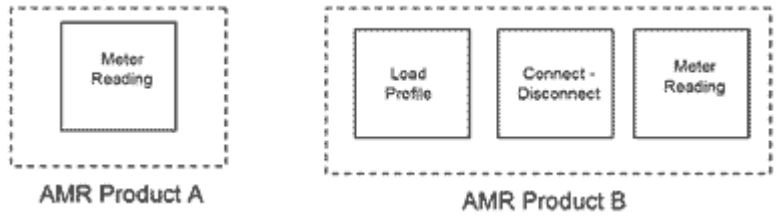
- name
- ampRating
- phases
- normalOpen
- inTransitTime
- switchOnCount
- switchOnDate
- aliasName
- description
- pathName

MultiSpeak V2.0 Process Model



MultiSpeak Functional Models

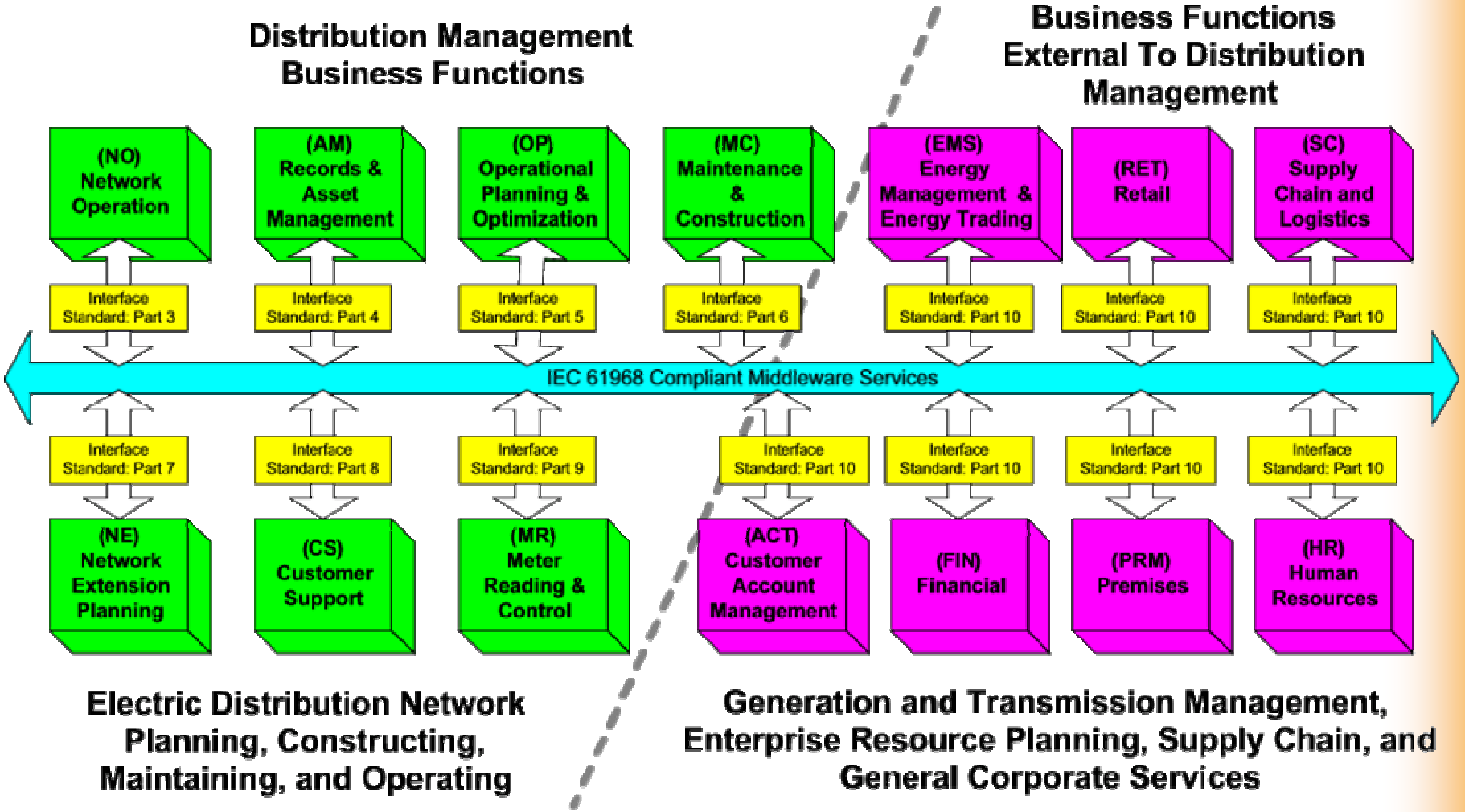
- Applications can be comprised of one or more functions
- Examples multi-function applications include:
  - Automated Meter Reading
  - Customer Information System
  - Geographic Information System
  - GIS Viewer
  - Interactive Voice Response
  - Outage Management



MultiSpeak Functions

- Customer Billing
- Connect/Disconnect
- Call Handling
- Customer Relationship Management
- Engineering Analysis
- Finance and Accounting
- Geographic Information Systems
- Load Management
- Load Profile
- Meter Reading
- Outage Analysis
- Outage Detection
- Static GIS Viewer
- Supervisory Control and Data Acquisition
- Automated Staking

IEC 61968 Interface Reference Model



MultiSpeak and IEC 61968 Messaging

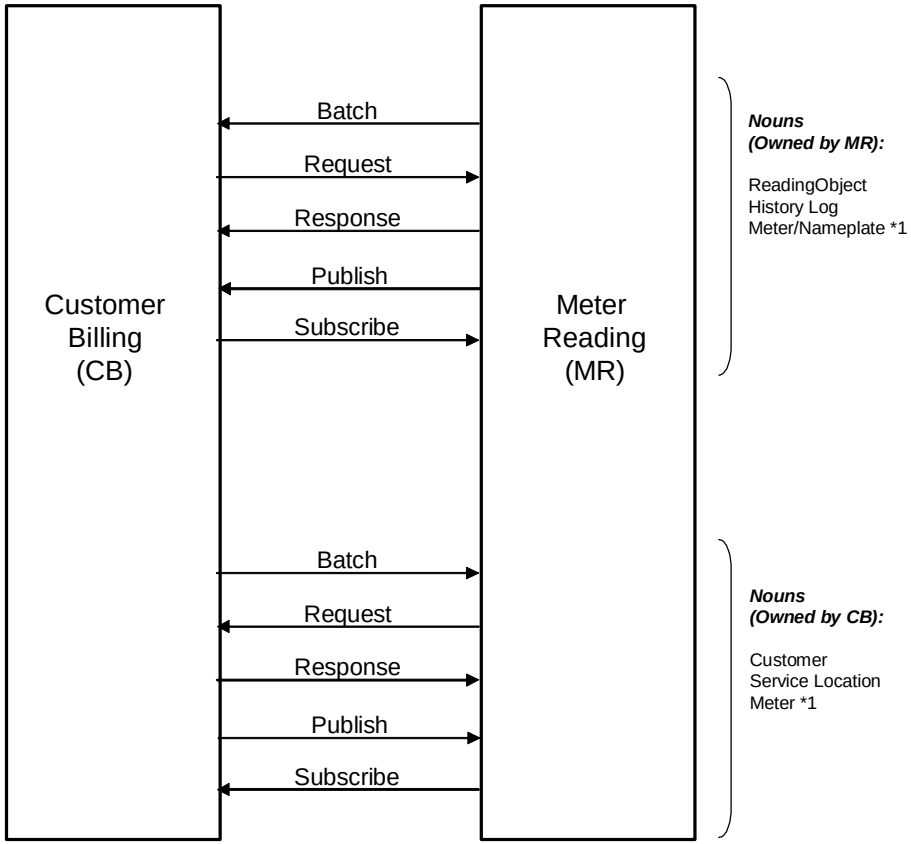
Both support these communication models:

- Batch
- Request/reply
- Publish/subscribe

IEC 61968-1 is transport independent, but MultiSpeak specifies:

- File-based transfers
- SOAP messages using HTTP
- Streamed over TCP/IP sockets connections directly between applications

MultiSpeak Integration Scenario



Notes:

1) The meter element is owned by the CB function;the nameplate child element of the meter element is owned by the MR function.

Example MultiSpeak Nouns

- crewActionEvent
- customerCall
- loadFlowResult
- outageDetectionEvent
- outageEvent
- equipmentList
- workOrder
- workTicket
- equipmentList
- employeeList
- meterReading
- timesheet
- customersAffectedbyOutage
- feederList
- genericLineFeatureList
- genericPointFeatureList

Example 61968 Nouns

- NetworkDataSet
- NetworkChangeSet
- LoadDataSet
- OperationalRestriction
- SafetyDocument
- SwitchingSchedule
- OutageReport
- EquipmentList
- TroubleTicket
- PlannedOutage
- OutageNotice
- Work
- Schedule
- Diagram
- AccessPermit
- OneCallRequest

MultiSpeak Verbs and Message Types

- NEW
- CHANGE
- DELETE
- REPLACE
- LINK
- UNLINK

- Response
- Request
- UnsubscribeNotice
- SubscriptionPing
- Ack
- Batch

IEC 61968 Verbs

Request/Reply:

- CREATE
- CHANGE
- CANCEL
- CLOSE
- DELETE
- GET
- SHOW
- REPLY

Publish/Subscribe:

- SUBSCRIBE
- UNSUBSCRIBE
- CREATED
- CHANGED
- CLOSED
- CANCELED
- DELETED

Example IEC 61968 Messages

Example Combinations of Verbs and Nouns:

- GetSwitchingSchedule
- SubscribeOutageReport
- CreatedOutageReport
- CreateOperationalRestriction
- ChangeOperationalRestriction
- CancelOperationalRestriction

MultiSpeak Message Header

```
<xsd:complexType name="MultiSpeakMessageHeader">
	<xsd:sequence>
		<xsd:element name="TimeStamp" type="xsd:dateTime" nillable="false"></xsd:element>
		<xsd:element name="DocumentID" type="xsd:string" nillable="false"></xsd:element>
		<xsd:element name="VendorApp" type="VendorApp" nillable="false"></xsd:element>
		<xsd:choice>
			<xsd:element ref="Response"/>
			<xsd:element ref="Request"/>
			<xsd:element name="UnsubscribeNotice" type="UnsubscribeNotice">
				<xsd:annotation>
					<xsd:documentation>This message is an unsolicited message sent by the publisher to the subscriber indicating that its subscription is about to be unilaterally dropped.  The purpose of this message is to drop ill-behaved applications or to permit orderly shutdown</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
			<xsd:element name="SubscriptionPing" type="SubscriptionPing">
				<xsd:annotation>
					<xsd:documentation>This message is used by the subscriber to "ping" the publisher or by the subscriber to ping the publisher to determine the status of a subscription.  Also called a "heartbeat" message.</xsd:documentation>
				</xsd:annotation>
			</xsd:element>
			<xsd:element ref="Ack"/>
			<xsd:element name="Batch" type="Batch"/>
		</xsd:choice>
	</xsd:sequence>
	<xsd:attribute name="Version" type="xsd:string" use="required"/>
	<xsd:attribute name="Username" type="xsd:string"/>
	<xsd:attribute name="Password" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="Batch">
	<xsd:sequence><xsd:element name="MultiSpeak" type="MultiSpeak" minOccurs="0"/></xsd:sequence>
</xsd:complexType>
<xsd:complexType name="VendorApp">
	<xsd:sequence>
		<xsd:element name="AppName" type="xsd:string"></xsd:element>
		<xsd:element name="AppVersion" type="xsd:string"></xsd:element>
		<xsd:element name="Company" type="xsd:string"></xsd:element>
		<xsd:element name="Function" type="xsd:string"></xsd:element>
	</xsd:sequence>
</xsd:complexType>
```

MultiSpeak Batch Document

```
<?xml version="1.0" ?>
<MultiSpeakMessageHeader Version="2.2" xmlns="http://www.multispeak.org/Schema/Version2.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        instance">
        <TimeStamp>2003-10-24T11:52:09-05:00</TimeStamp>
        <DocumentID>Show14</DocumentID>
        <VendorApp>
                <AppName>MultiSpeak RTI Server</AppName>
                <AppVersion>0.1</AppVersion>
                <Company>LiveData, Inc.</Company>
                <Function>All</Function>
        </VendorApp>
        <Batch>
                <MultiSpeak documentType="dump">
                        <customer objectID="123456789000001" utility="UtilityA">
                                <lastName>Peters</lastName>
                                <firstName>Stanley</firstName>
                                <mName>K</mName>
                                <dBAName>Business A</dBAName>
                                <homeAc>717</homeAc>
                                <homePhone>4522345</homePhone>
                                <dayAc>717</dayAc>
                                <dayPhone>4522345</dayPhone>
                                <billAddr1>1011 Schaub Drive</billAddr1>
                                <billAddr2>Suite 200</billAddr2>
                                <billCity>Raleigh</billCity>
                                <billState>NC</billState>
                                <billZip>27606</billZip>
                        </customer>
.....
```

MultiSpeak customerCall

```
<xsd:element name="customerCall" type="customerCall" substitutionGroup="mspObject"/>
<xsd:complexType name="customerCall">
        <xsd:complexContent>
                <xsd:extension base="mspObject">
                        <xsd:sequence>
                                <xsd:element name="servLoc" type="objectID" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="custID" type="custID" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="accountNumber" type="accountNumber" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="callTime" type="mspDate" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="description" type="description" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="typeCode" type="typeCode" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="source" type="callSource" minOccurs="0">
                                </xsd:element>
                                <xsd:element name="takenBy" type="xsd:string" minOccurs="0">
                                </xsd:element>
                        </xsd:sequence>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>
```

(defined in MultiSpeak.xsd)

IEC 61968 Message Headers

```
<xsd:element name="ControlAreaGroup" type="cg:ControlAreaGroupType">
</xsd:element>
<xsd:complexType name="ControlAreaGroupType">
        <xsd:sequence>
                <xsd:element name="Verb">
                        <xsd:simpleType>
                                <xsd:restriction base="xsd:string">
                                        <xsd:enumeration value="cancel"/>
                                        <xsd:enumeration value="canceled"/>
                                        <xsd:enumeration value="changed"/>
                                        <xsd:enumeration value="close"/>
                                        <xsd:enumeration value="closed"/>
                                        <xsd:enumeration value="create"/>
                                        <xsd:enumeration value="created"/>
                                        <xsd:enumeration value="delete"/>
                                        <xsd:enumeration value="deleted"/>
                                        <xsd:enumeration value="get"/>
                                        <xsd:enumeration value="reply"/>
                                        <xsd:enumeration value="show"/>
                                        <xsd:enumeration value="subscribe"/>
                                        <xsd:enumeration value="unsubscribe"/>
                                </xsd:restriction>
                        </xsd:simpleType>
                </xsd:element>
                <xsd:element name="Noun" type="xsd:string">
                </xsd:element>
                <xsd:element name="Revision" type="xsd:string" minOccurs="0">
                </xsd:element>
                <xsd:element name="TimeDate" type="cim:AbsoluteDateTime" minOccurs="0">
                </xsd:element>
        </xsd:sequence>
</xsd:complexType>
```

IEC 61968 Planned Outage

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="PlannedOutage" xmlns:po="PlannedOutage" xmlns:oag="http://www.openapplications.org/oagis" xmlns:cs="cimSegments"
        xmlns:cim="cimBase" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:cg="cimGroups" elementFormDefault="unqualified"
        attributeFormDefault="unqualified">
        <xsd:import namespace="cimGroups" schemaLocation="cimGroups.xsd"/>
        <xsd:element name="PlannedOutage">
                <xsd:annotation><xsd:documentation>A planned outage involves network operations which will affect the supply of power to
        customers.</xsd:documentation></xsd:annotation>
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref="cg:ControlAreaGroup"/>
                                <xsd:element name="MessagePayload" type="po:PlannedOutageDataArea">
                                        <xsd:annotation>
                                                <xsd:documentation>Message Data Area</xsd:documentation>
                                        </xsd:annotation>
                                </xsd:element>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:complexType name="PlannedOutageDataArea">
                <xsd:annotation><xsd:documentation>Message data area.</xsd:documentation></xsd:annotation>
                <xsd:sequence>
                        <xsd:annotation><xsd:documentation>Information relative to a planned outage.  Note that details of each outage step can be
        exchanged through the OutageRecord message type.</xsd:documentation></xsd:annotation>
                        <xsd:element ref="cs:PlannedOutageSeg"/>
                        <xsd:element ref="cs:OutageScheduleSeg" minOccurs="0"/>
                        <xsd:element ref="cs:EquipmentListSeg" minOccurs="0"/>
                        <xsd:element ref="cs:OutageRecordSeg" minOccurs="0"/>
                        <xsd:element ref="cs:CustomerListSeg" minOccurs="0"/>
                        <xsd:element ref="cs:ActivityRecordSeg" minOccurs="0" maxOccurs="unbounded"/>
                </xsd:sequence>
        </xsd:complexType>
</xsd:schema>
```

Maintenance/Use of Models

MultiSpeak:

- Model maintained using XML Spy as Multispeak.xsd
- Web pages generated using XML Spy
- Messages defined using XML Spy, with XML Schema for each flow to defined required fields
- Example batch files provided for compliance testing

CIM:

- Source model maintained as an .mdl and .cat files using Rational Rose
- Web pages with graphics generated from Rose model
- XML Schemas generated from Rational Rose
- RDF Schema generated from Rational Rose
- XML Spy used to define messages with references to CIM, OAG and GML .xsd files

Summary

- MultiSpeak and CIM models are different, where some portions of CIM parallel MultiSpeak models
- MultiSpeak and CIM efforts used modeling tools differently
- MultiSpeak has functional overlap with IEC 61968
- IEC 61968 is transport independent, while MultiSpeak defines SOAP, sockets and files
- Both use XML Schema to define message payloads
- Message headers can be readily mapped between MultiSpeak and IEC 61968
- Mapping of message content (payloads) between the two is more complex

More Information

- IEC international standards can be obtained from the IEC web site at http://www.iec.ch
- MultiSpeak specifications can be downloaded from http://www.multispeak.org
- CIM model is posted on the WG13/14 web site: http://www.cimuser.com
- US utilities and companies can join the WG14 effort by contacting Scott Neumann at sneumann@uisol.com
- For participating in MultiSpeak, contact Gary McNaughton at gmcnaughton@frontier.net

Thanks to Gary McNaughton for review and comments