Model Driven Transformation

Alan McMorran B.Eng Ph.D
Susan Rudd B.Sc Ph.D

Open Grid Systems

Modelling Data

Open Grid Systems

What is Information Modelling

- "An **information model** is a representation of **concepts**, **relationships, constraints, rules,** and **operations** to specify data **semantics** for a chosen **domain** of discourse"

  -- Y. Tina Lee, Information Modeling: From Design to Implementation, NIST, 1999

- An information model defines the **structure** of data independently of a particular implementation technology

- **File formats** and **database schemas** can be **derived** from the information model

Open Grid Systems

Defining Models

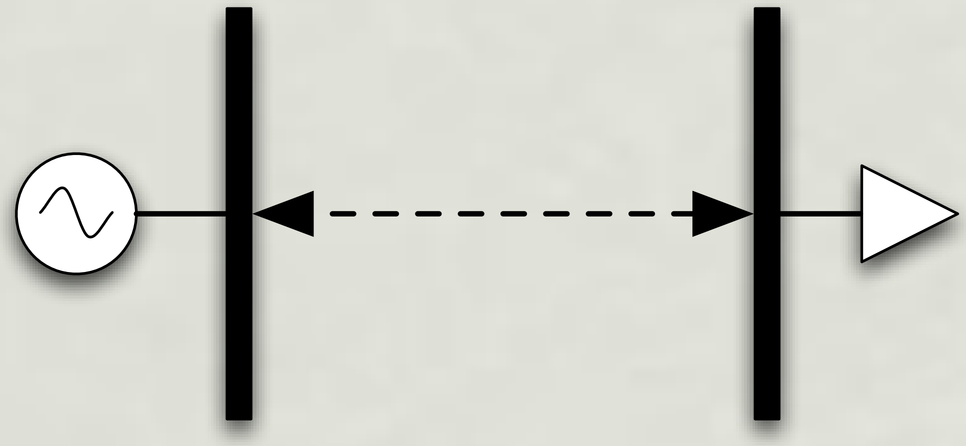
- In some cases an information model will **already exist** e.g. IEC61970/61968, MultiSpeak, IEC 61850 or an internal Enterprise Semantic Model
- Sometimes the information model is **implicit to the format** and can be **derived automatically** e.g. database schema or XML Schema Definitions (XSD)
- Other times you need to **reverse-engineer** the information model from the format itself e.g. CSV files or column-oriented text formats

Open Grid Systems

Simple Example

- It is relatively easy to define a simple data format that mirrors your current requirements for an application or project

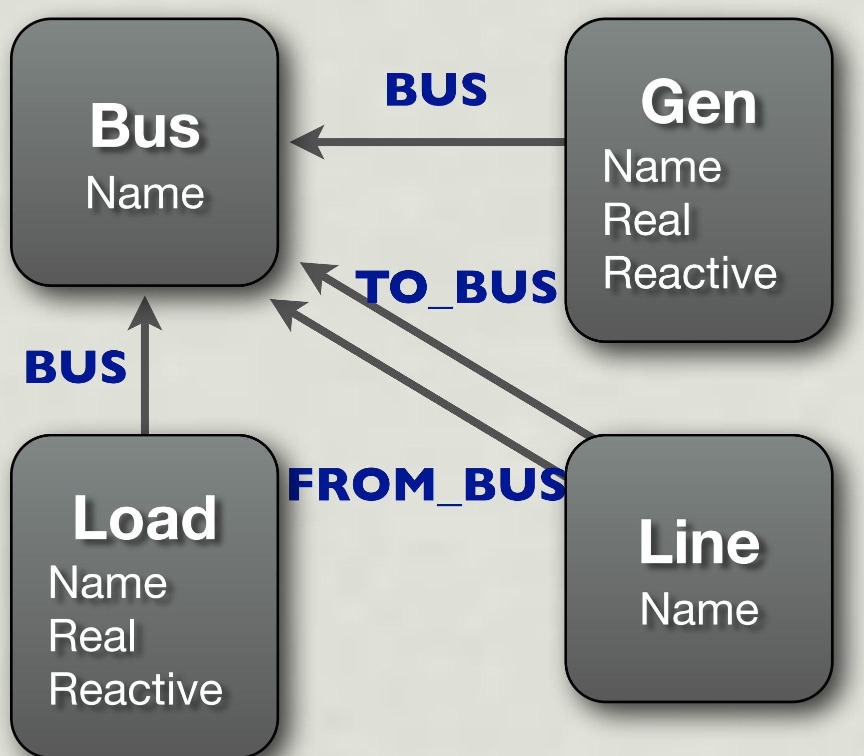
```
#BUS NAME
 1   GLAZ
 2   LHRX
#GEN NAME REAL REACTIVE BUS
 1   G1   6.2  1.2      1
#LOAD NAME REAL REACTIVE BUS
 1    LD1  5.3  0.8      2
#LINE NAME TO_BUS FROM_BUS
 1    GL1  1      2
```



Open Grid Systems

Structure vs Format

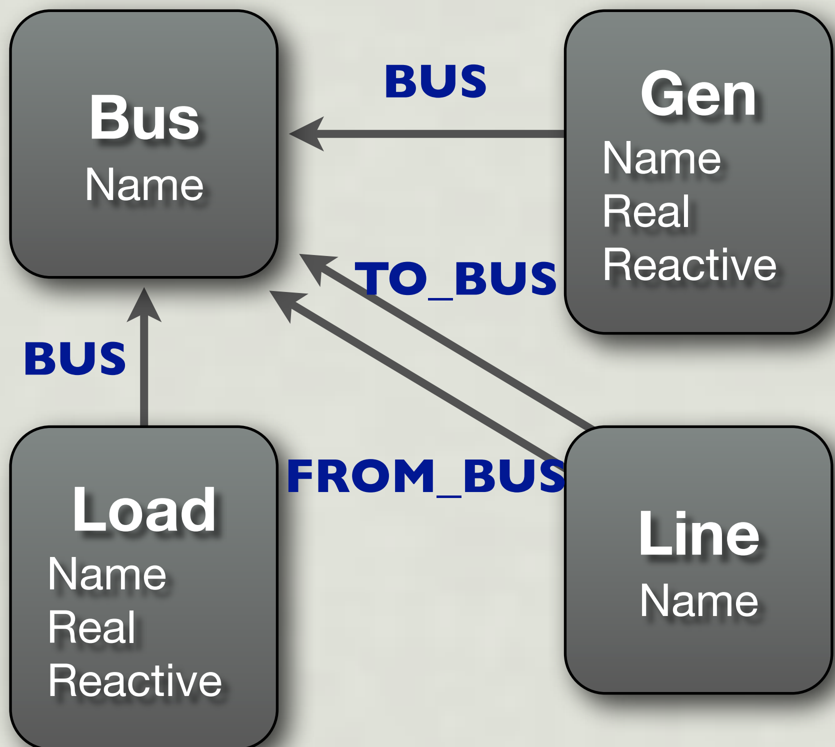
```
#BUS NAME
 1   GLAZ
 2   LHRX
#GEN NAME REAL REACTIVE BUS
 1   G1   6.2  1.2      1
#LOAD NAME REAL REACTIVE BUS
 1    LD1  5.3  0.8      2
#LINE NAME TO_BUS FROM_BUS
 1    GL1  1      2
```



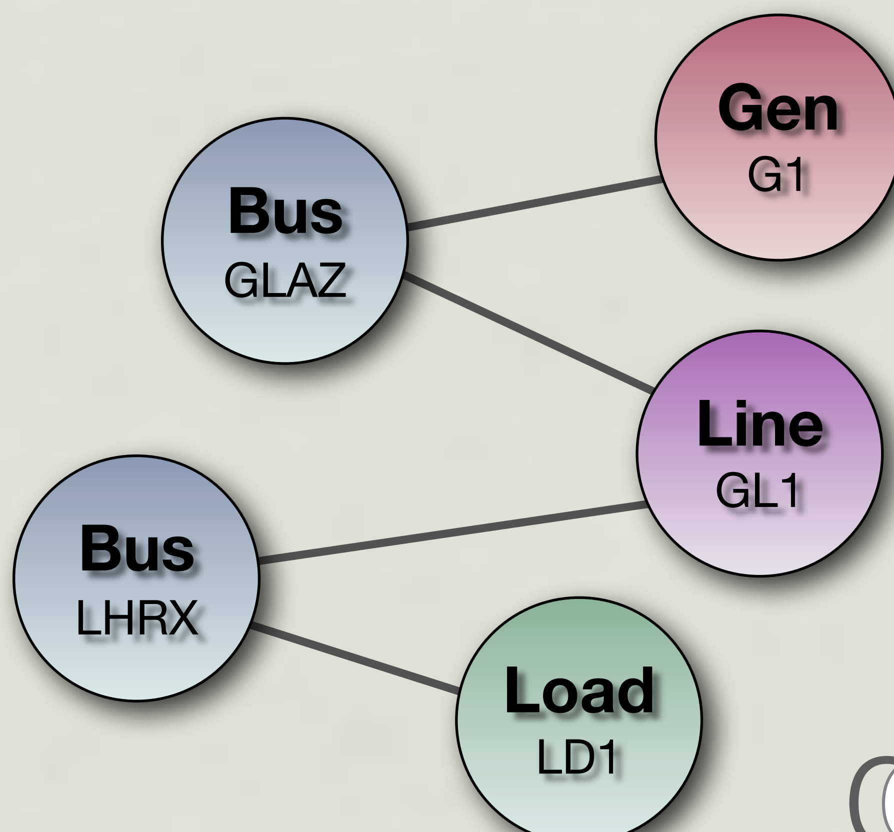
- The structure of this data is **independent** of one particular format
- There are 4 distinct types of data: **Bus**, **Gen**, **Load**, and **Line**
- There are also **relationships** between the different types
- The structure is **abstract**, the format is just how we serialise it

Open Grid Systems

One Structure, Multiple Formats

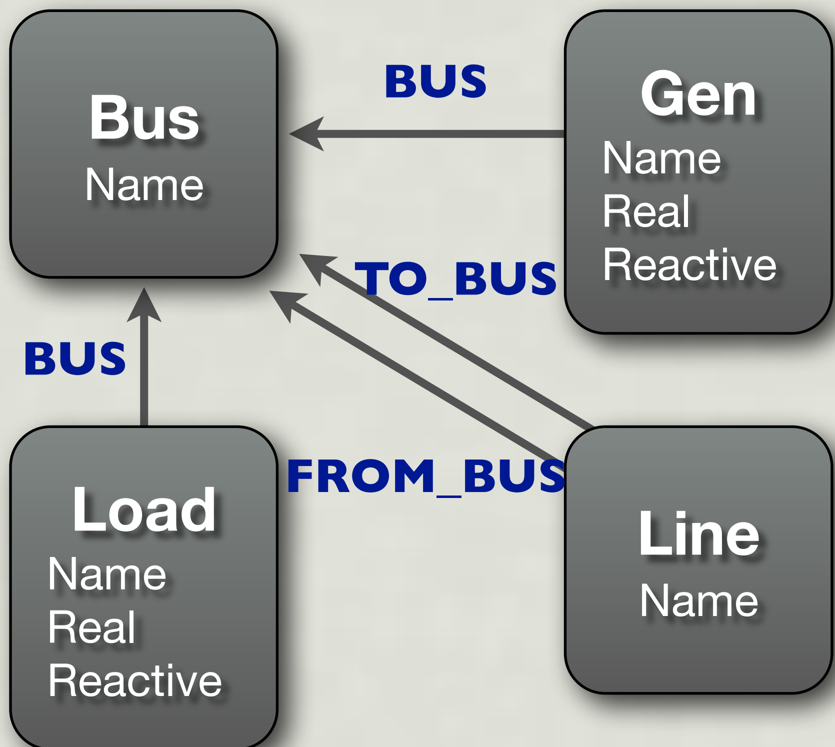



```
#BUS NAME
 1   GLAZ
 2   LHRX
#GEN NAME REAL REACTIVE BUS
 1   G1   6.2  1.2      1
#LOAD NAME REAL REACTIVE BUS
 1    LD1  5.3  0.8      2
#LINE NAME TO_BUS FROM_BUS
 1    GL1  1      2
```

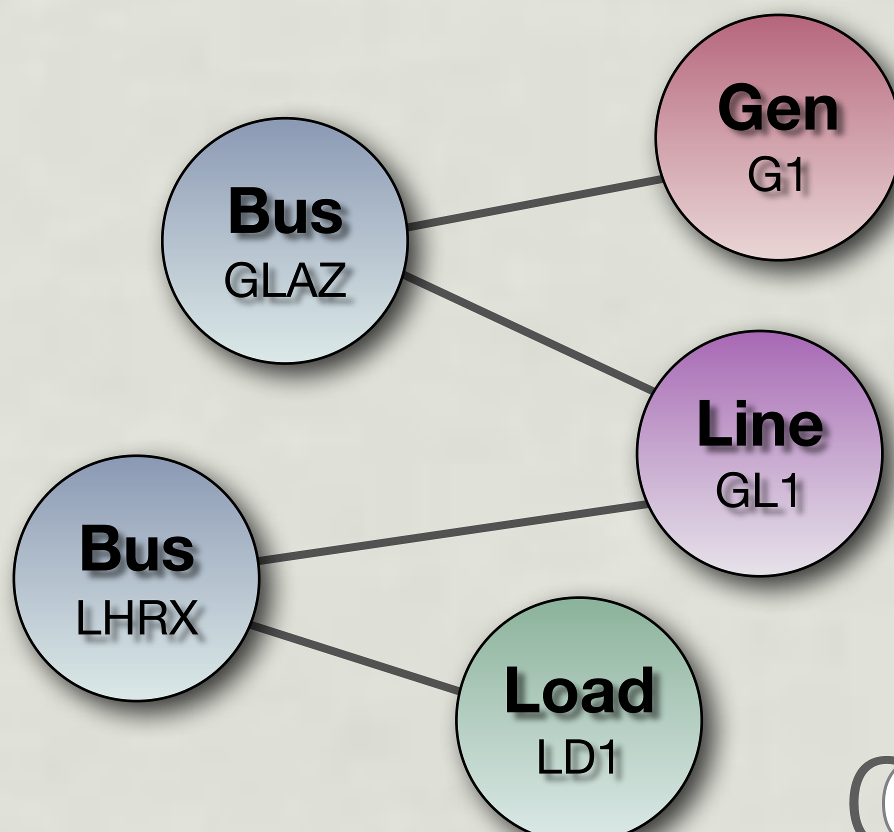


Open Grid Systems

One Structure, Multiple Formats

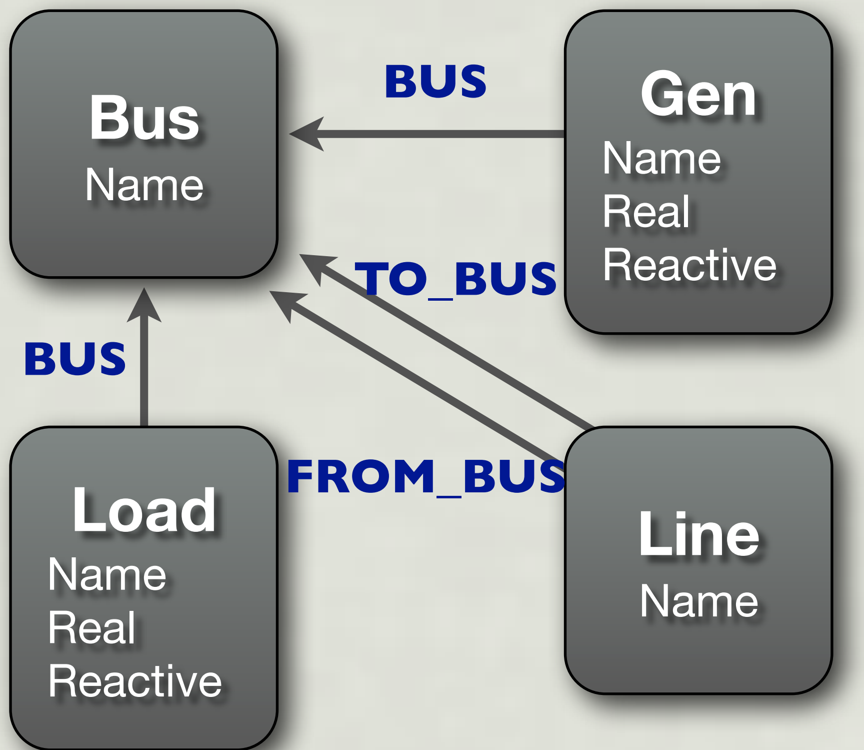



```
<BUS NAME="GLAZ">
    <GEN NAME="G1" P="6.2" Q="1.2/>
</BUS>
<BUS NAME="LHRX">
    <LOAD NAME="LD1" P="5.3" Q="0.8/>
</BUS>
<LINE NAME="GL1" TO="//@BUS.1" FROM="//@BUS.2"/>
```

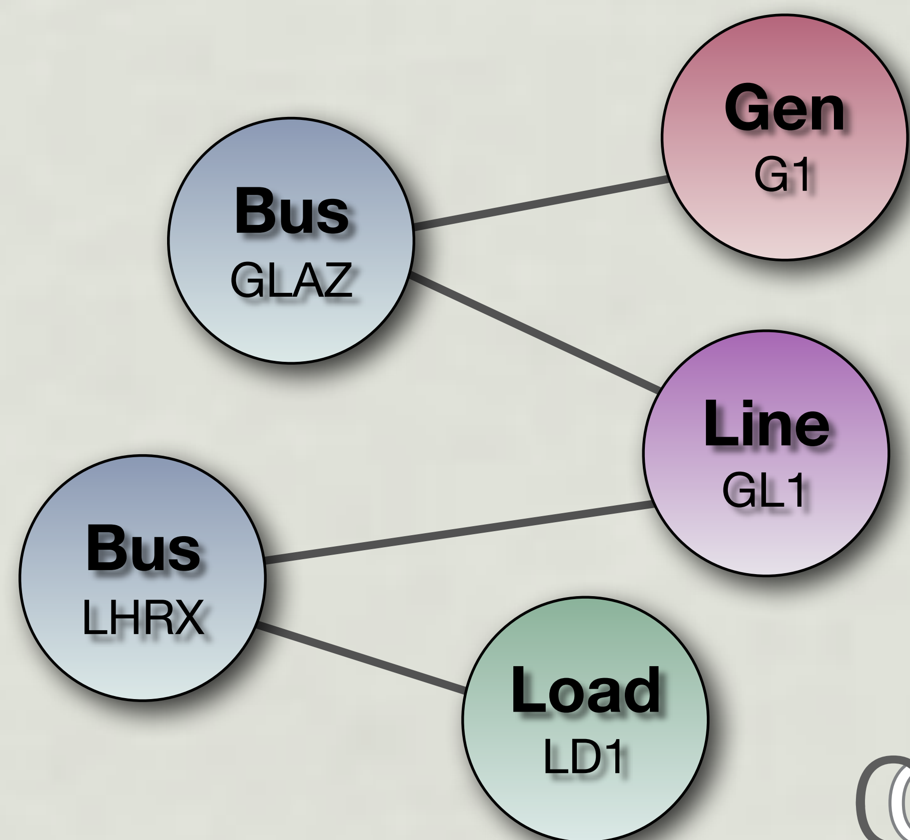


Open Grid Systems

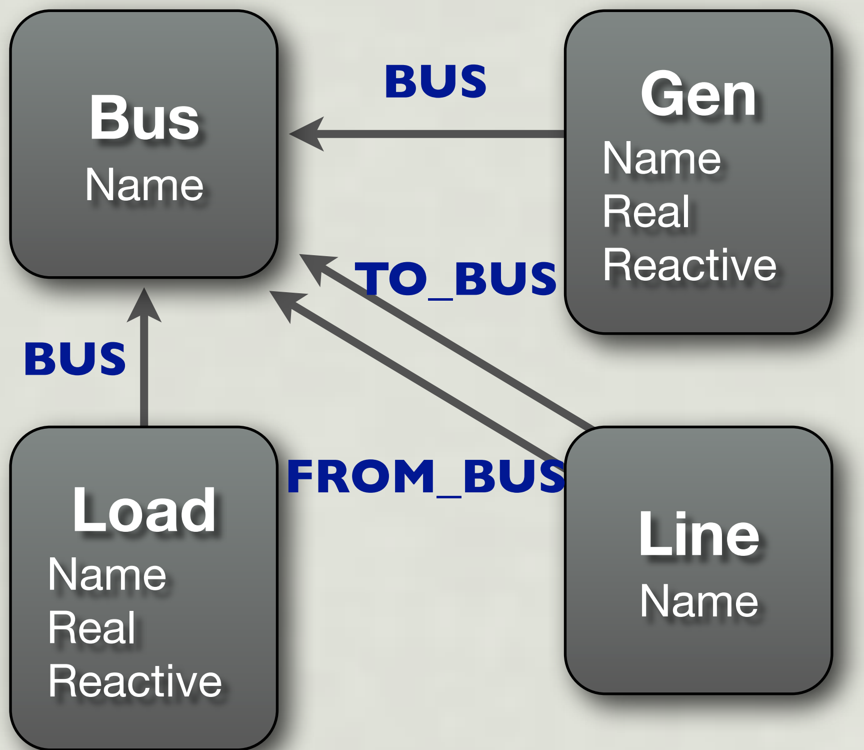
One Structure, Multiple Formats




```
BUS, 1, GLAZ
BUS, 2, LHRX
GEN, 1, G1, 6.2, 1.2, 1
LOAD, 1, LD1, 5.3, 0.8, 2
LINE, 1, GL1, 1,2
```

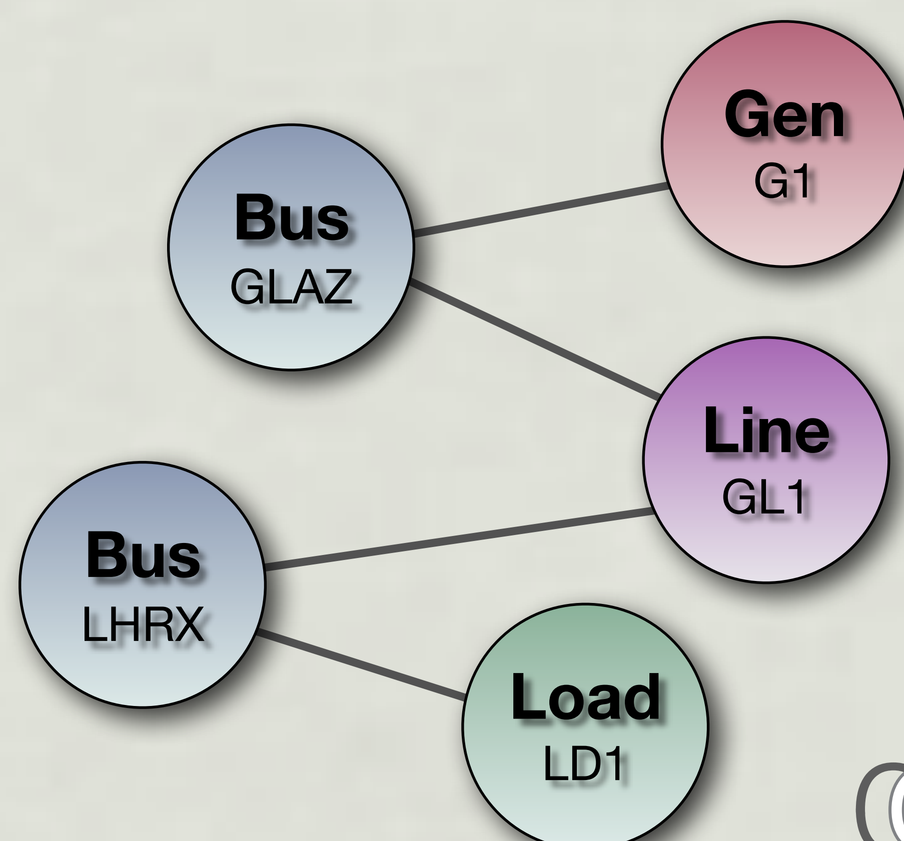


One Structure, Multiple Formats



4CK?Q*-V?K?M?}?0???1
??B?I?GB?ᵛZF?@??@??1>AC|?S??L????Y 菲??? ???|G_??
D,?V???$R?

.?:???



Open Grid Systems

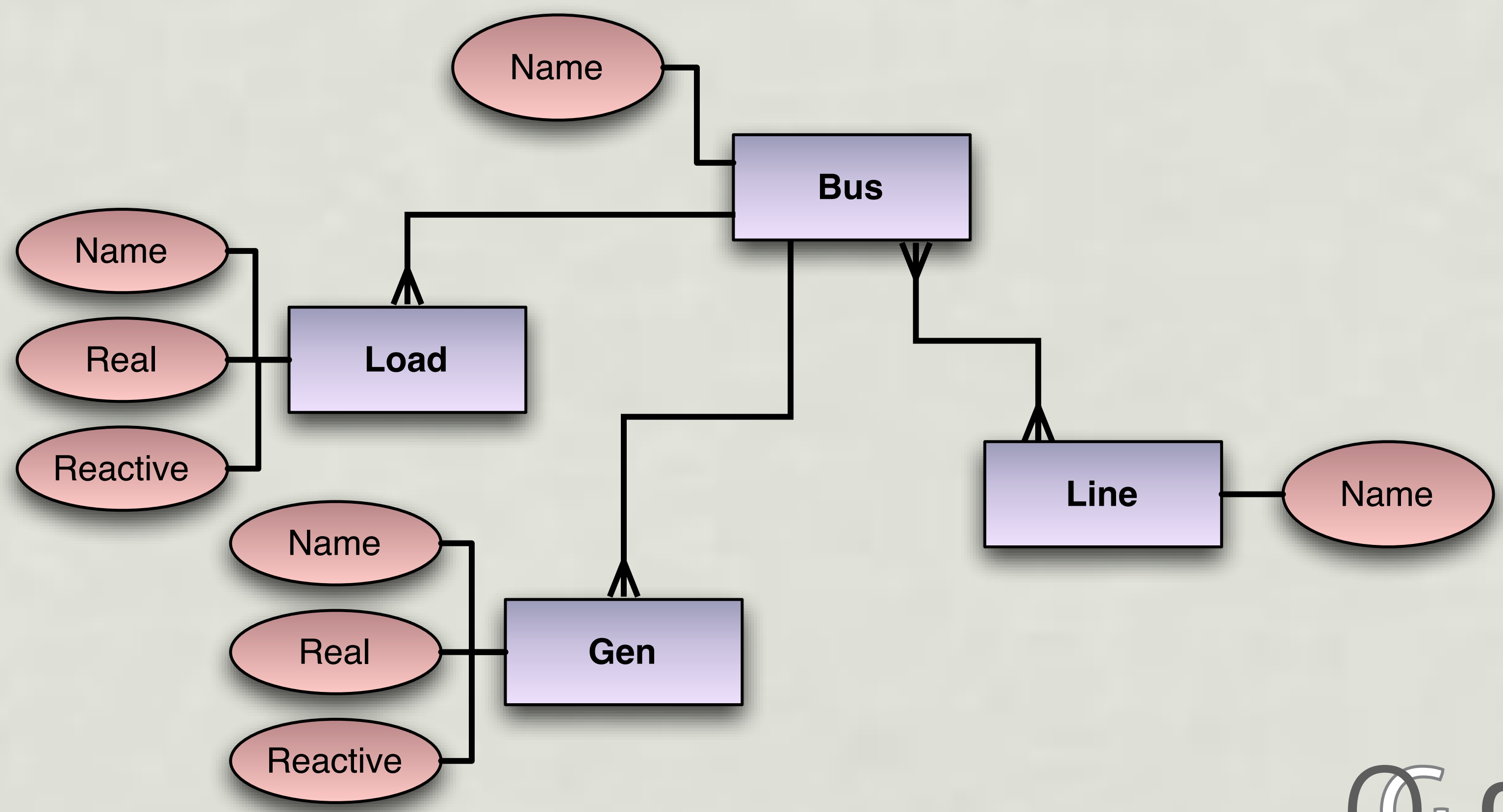
One Structure, Multiple Formats

- It is not uncommon to get hung up on one particular format of data, ignoring the importance of the **structure**
- Structure will define how easy it is for others to **interpret**, **understand** and **use** your data
- Initial modelling of this data should (in an ideal world) be done **before** any software is bought/written or interfaces defined
- There are different languages in which you can model data

Open Grid Systems

Modelling Data

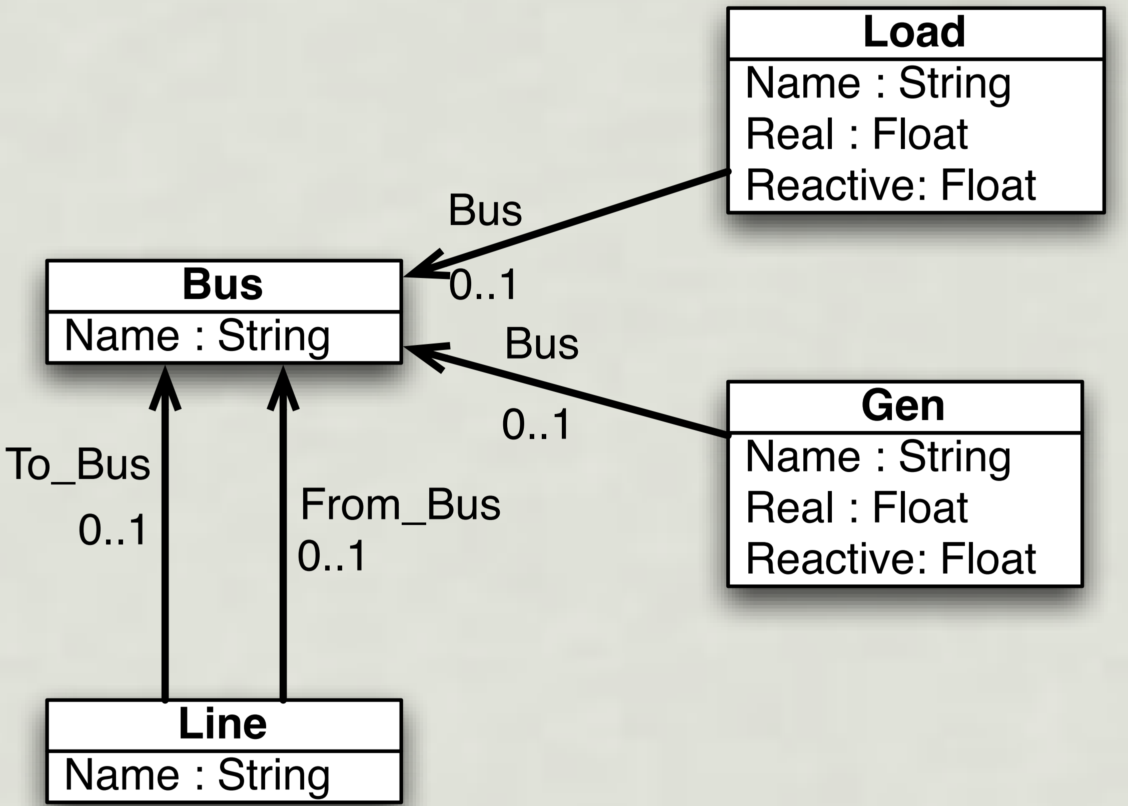
- How you model data will depend on how complex your data requirements are (and what you're comfortable with!)
- A simple entity relationship model shows entities (types), their attributes and the relationships between entities



Open Grid Systems

Modelling Data

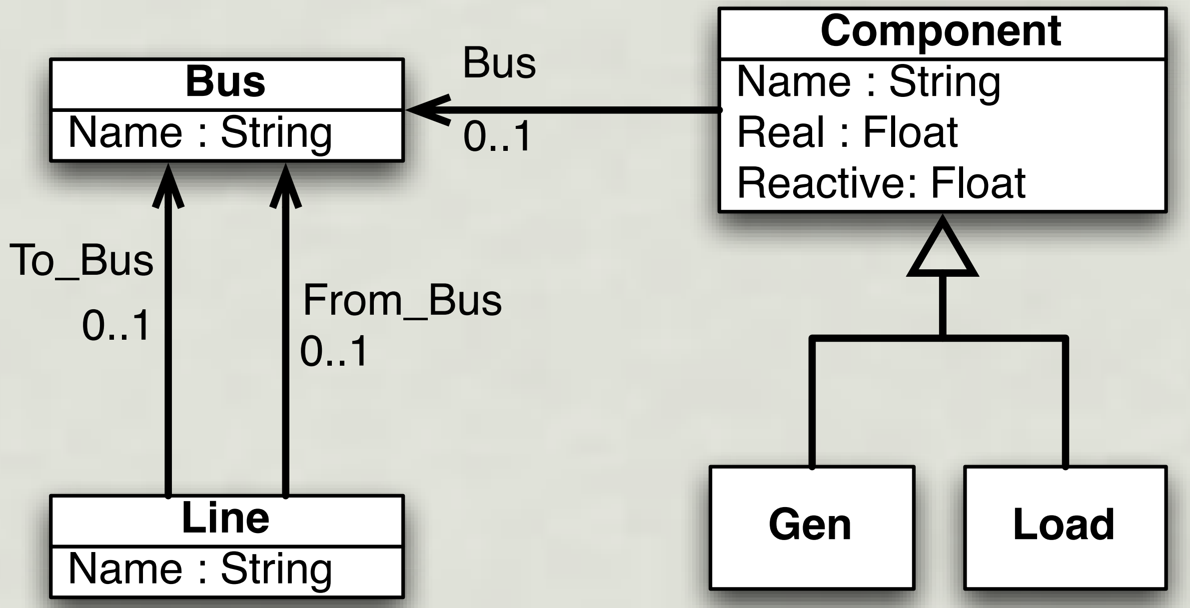
- The Unified Modelling Language (UML) provides another language that, amongst other things, include support for class inheritance



Open Grid Systems

Modelling Data

- The Unified Modelling Language (UML) provides another language that, amongst other things, include support for class inheritance



Open Grid Systems

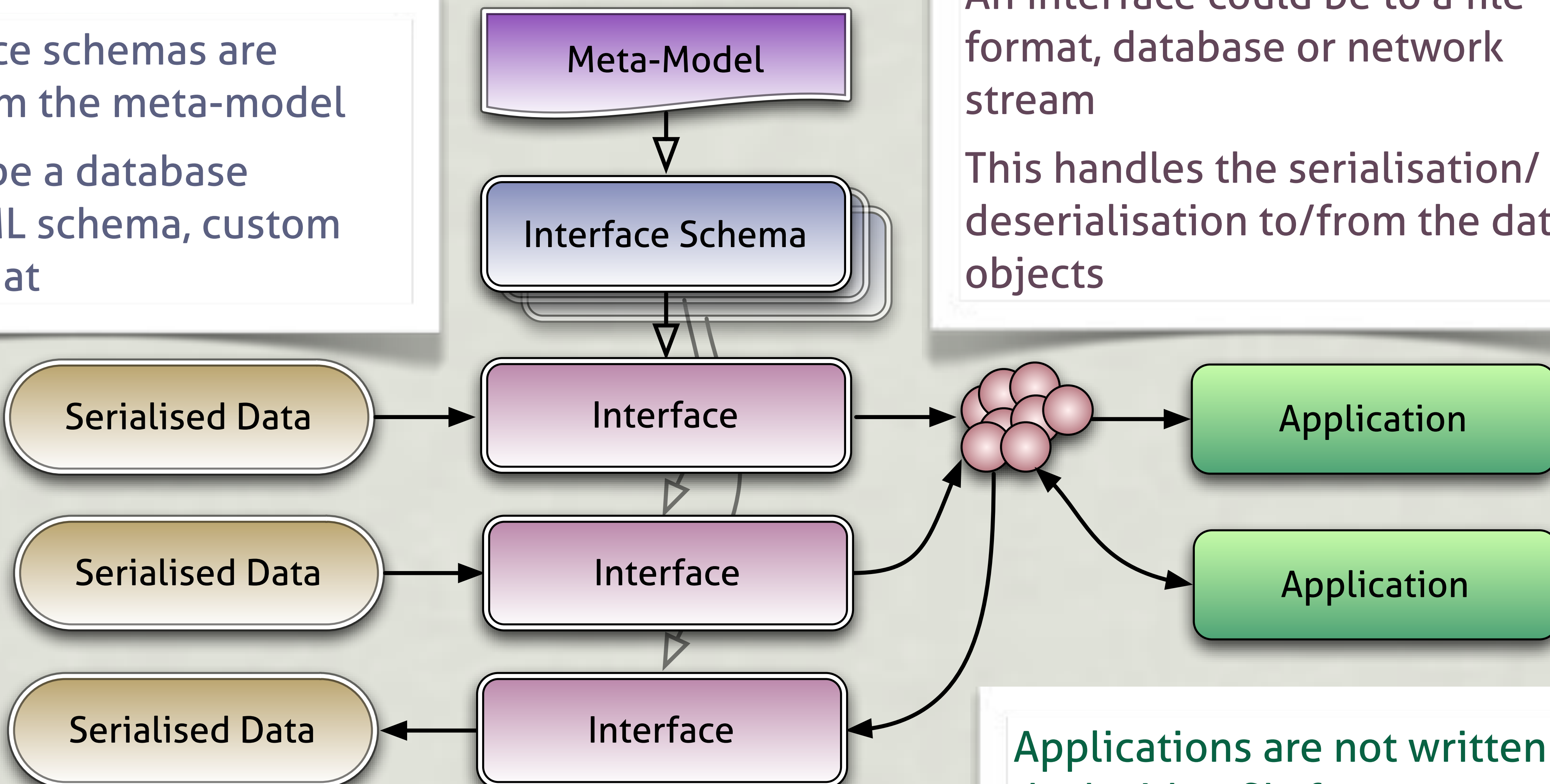
Modelling Components

The interface schemas are derived from the meta-model

This could be a database schema, XML schema, custom binary format

An interface could be to a file format, database or network stream

This handles the serialisation/ deserialisation to/from the data objects



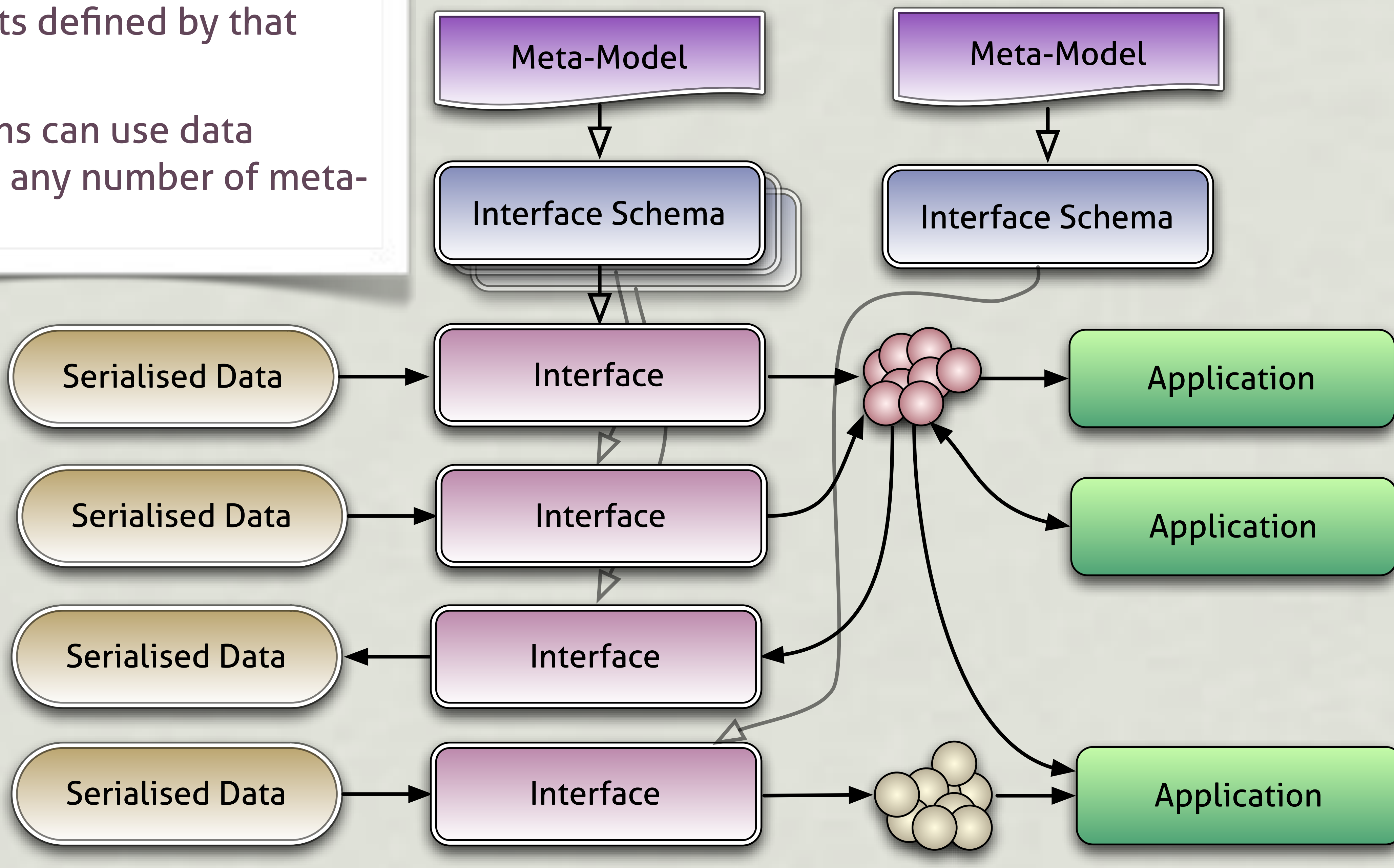
Applications are not written to deal with a *file format*

They run against the data objects, independent of its source/target

Modelling Components

Different Meta-Models produce data objects defined by that model

Applications can use data defined by any number of meta-models



Open Grid Systems

Standards

- Data **mapping** and **transformation** is one of the major issues for **integration** of data in the power industry
- Even with a model defining the data, requiring applications to deal with multiple overlapping models does not scale well
- Given that it can be a complex and time-consuming process it makes sense to have your data mapped to a single, **Common Information Model**
- This the **Holy Grail** for systems integration...

Open Grid Systems

Modelling in Eclipse

Eclipse Modelling Framework and OMG Standards

Open Grid Systems

Eclipse

- Eclipse is an **open-source**, **Java**-based Integrated Development Environment (IDE)
- Originally developed by **IBM** is has grown from being purely an IDE into a powerful platform on which applications can be developed
- Eclipse uses the **OSGi** modular Java runtime to provide a *plug-in* architecture
- Eclipse runs on all major operating systems including Windows, Linux and MacOS

Open Grid Systems

EMF

- The Eclipse Modelling Framework (EMF) provides a modelling framework and code-generation tools for Java
- The EMF framework provides standard APIs and forms the foundation of a number of Eclipse Modelling Tools for:
    - Model to Text
    - Domain Specific Languages
    - Validation
    - Transformation

Open Grid Systems

EMF Components

- The **Ecore** schema format defines the meta-model in EMF and is based on the Object Management Group (OMG) Meta Object Facility (MOF) format
- The Ecore schema itself is defined in Ecore so the schemas can be processed using model-driven tools!
- The **ResourceImpl** classes and associated registries enable data (de)serialisation and automatic discovery of interfaces for formats and protocols

XML Schema Support

- Since the meta-model in **Ecore** is itself a model, tools have been developed to derive an **interface schema** from the **meta-model** or to derive a meta-model from an existing **interface schema** (e.g. XSD)
- EMF includes tools to import existing **XSDs** and automatically create the **Ecore** and a corresponding **XMLResourceImpl**
- This allows XML messages defined by that XSD to be read and the corresponding **EObjects** to be created (and **EObjects** defined by that meta-model to be written as XML Messages)

Open Grid Systems

Extending EMF

- Using this architecture, model-driven support for the CIM and related standards can be added to EMF
- The CIM model is in UML and can be directly translated into **Ecore** including all classes, attributes and associations
- A CIM profile can be translated into either a sub-set of the **Ecore** model or as restrictions to the overall model
- The **IEC61970-552** RDF serialisation format is implemented as an **RDFResourceImpl**

EMF API

- The EMF code-generation provides Java code that can also be used as a normal API:

```
Breaker breaker = WiresFactory.eINSTANCE.createBreaker();
breaker.setName("My Breaker");
Terminal terminal = CoreFactory.eINSTANCE.createTerminal();
terminal.setName("My Terminal");
terminal.setConductingEquipment(breaker);

SvPowerFlow pf = StateVariablesFactory.eINSTANCE.createSvPowerFlow();
pf.setP(2.3f);
pf.setQ(0.9f);
pf.setTerminal(terminal);
```

- Or *reflectively*

```
terminal.eSet(CorePackage.Literals.TERMINAL__CONDUCTING_EQUIPMENT,
breaker);
```

Open Grid Systems

Reflective Access

- The reflective API allows software to dynamically discover the properties of the **EObject** (the format agnostic data object in EMF)

```java
public void print(EObject obj){
    // Determine our objects's EMF Class
    EClass cls = obj.eClass();
    System.out.println("Class: "+cls.getName());
    // Find all the native features of this EClass
    Collection<EStructuralFeature> features = cls.getEStructuralFeatures();
    // Loop through the features
    for (EStructuralFeature feature : features){
        // Check if the feature is an Attribute or Reference (Association)
        if (feature instanceof EAttribute)
            System.out.print("Attribute: ");
        else if (feature instanceof EReference)
            System.out.print("Reference: ");
        // Print out the name and type of the feature
        System.out.print(feature.getName() + " of type "+feature.getEType().getName());
        System.out.println(" = "+obj.eGet(feature));
    }
}
```

Reflective Access

- If this code is run against a CIM **PowerTransformer** object the output is:

```
Class: PowerTransformer
Reference: TransformerWindings of type TransformerWinding = []
Reference: HeatExchanger of type HeatExchanger = null
Attribute: magSatFlux of type PerCent = 0.0
Attribute: magBaseU of type Voltage = 0.0
Attribute: bmagSat of type PerCent = 0.0
```

- Software can be written for a specific, known meta-model and use the normal API for data access
- Model-agnostic software can automatically determine the structure from the data and access reflectively

Open Grid Systems

Projects Using EMF

- A number of Eclipse projects use EMF:
- The Eclipse implementation of the **Object Constraint Language** (OCL) uses EMF and the Ecore format as the model definition
- **QVT** (Operational) and **ATL** are model-driven transformation languages built on EMF
- GMF is the **Graphical Modelling Framework** that extends the Eclipse Graphical Editing Framework to provide model-driven graphical editors

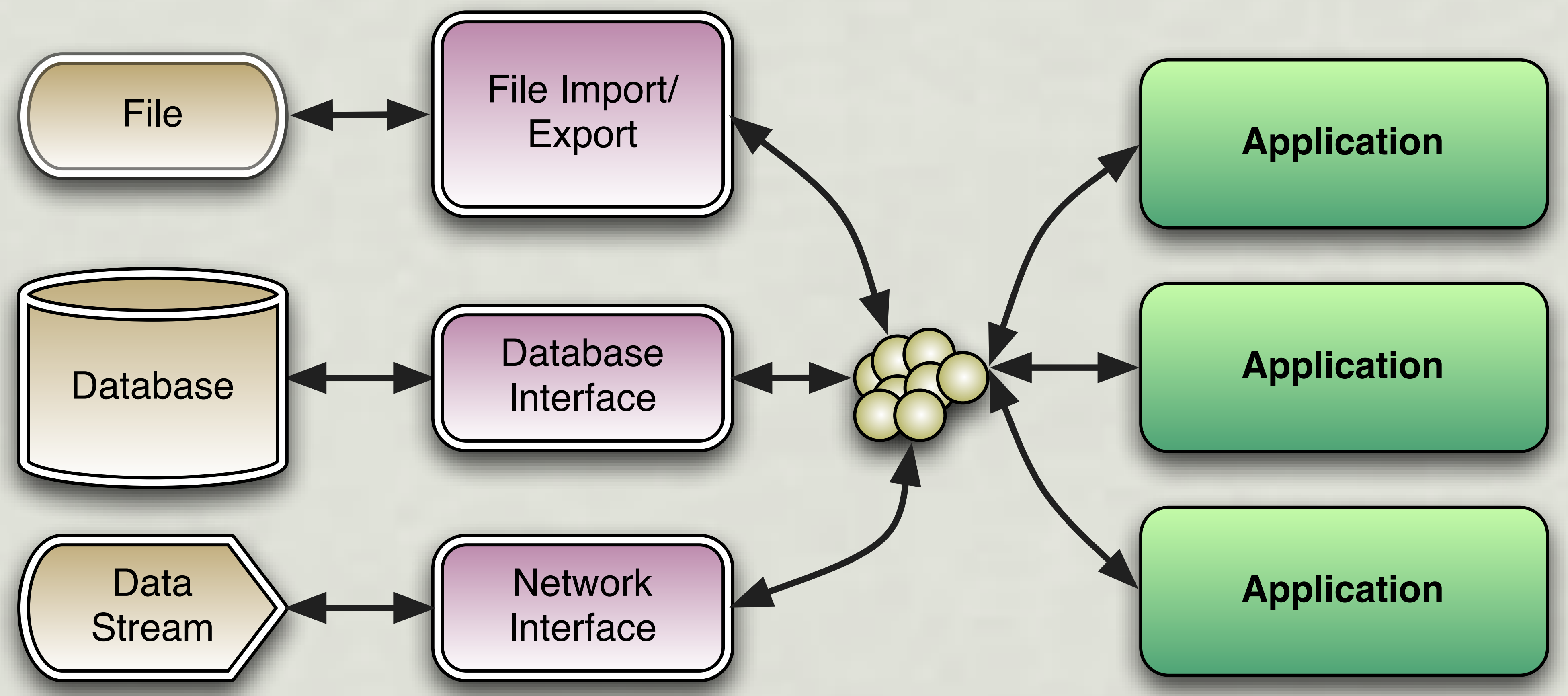
Model Driven Transformation

Data mapping and translation at the structural level

Open Grid Systems

Model-Driven Engineering

- Managing data at the model level allows the structure to define functionality
- Applications are written to deal with **data** irrespective of where it is stored and in what format



Open Grid Systems

Transformation

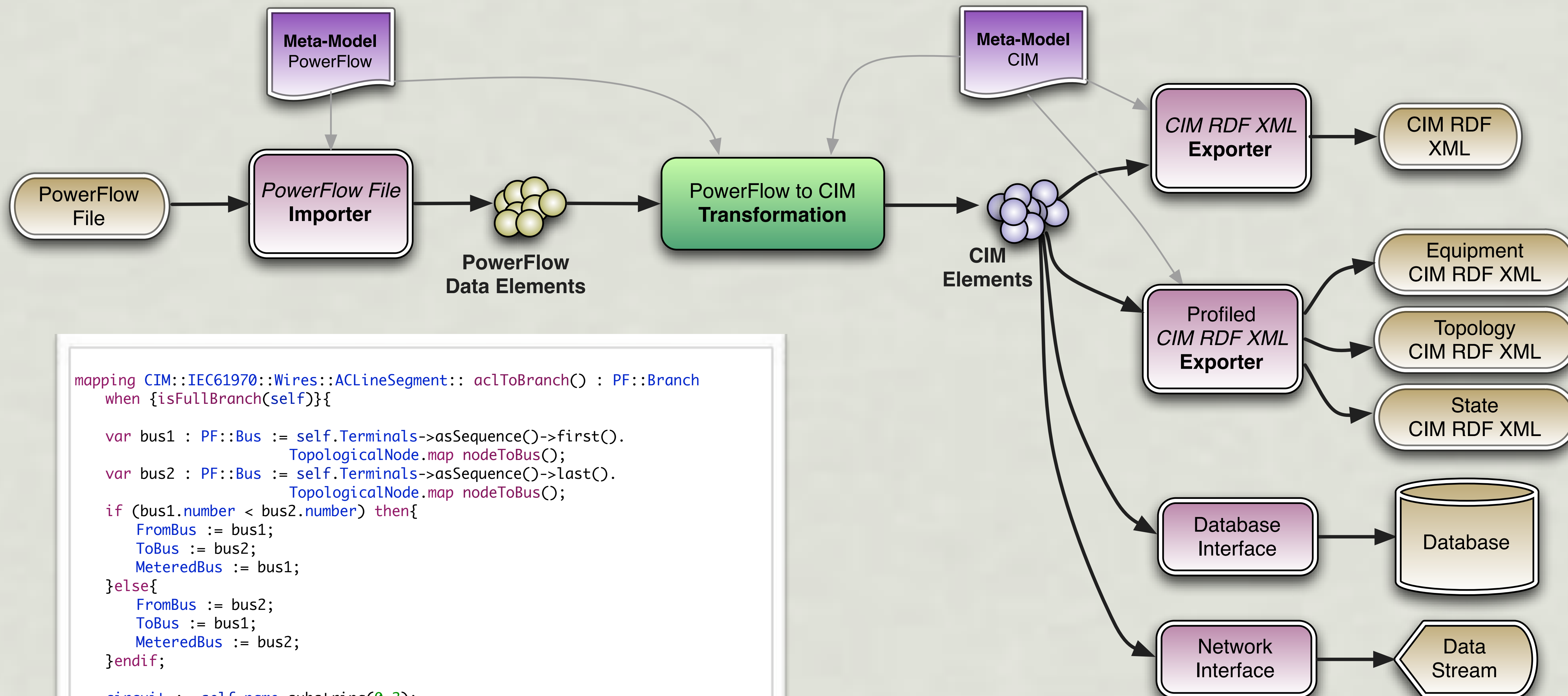
- In an ideal world all data within an enterprise will be defined by a single common information model
- Alas the world is not ideal and there is a need to **transform data** between **structures** and formats
- This may be to support legacy application data or even to harmonise data between standards
- **Model-Driven Transformation** defines the data mappings at the **meta-model level**, independent of the source and target formats

Open Grid Systems

Transform Definitions

- Transformations are written against the model and there are a number of model-driven transformation languages including **QVT** and **ATL**
- The same transformation can be used with data coming from web services, databases, files or any other data source as they run against the **data objects**
- Interfaces can be model-agnostic (e.g. RDF XML) or model dependent (e.g. power-flow formats)
- Multiple interfaces can be used to get data in the same structure to/from different formats

Transformation Workflow




```
mapping CIM::IEC61970::Wires::ACLineSegment:: aclToBranch() : PF::Branch
   when {isFullBranch(self)}{

   var bus1 : PF::Bus := self.Terminals->asSequence()->first().
                         TopologicalNode.map nodeToBus();
   var bus2 : PF::Bus := self.Terminals->asSequence()->last().
                         TopologicalNode.map nodeToBus();
   if (bus1.number < bus2.number) then{
      FromBus := bus1;
      ToBus := bus2;
      MeteredBus := bus1;
   }else{
      FromBus := bus2;
      ToBus := bus1;
      MeteredBus := bus2;
   }endif;

   circuit := self.name.substring(0,2);
   R := self.r.toPerUnit();
   X := self.x.toPerUnit();
   B := self.bch.toPerUnit();
   length := self.length;
}
```

Open Grid Systems

Case Study

Converting CIM RDF XML into PSS®E RAW

Open Grid Systems

Example: CIM to PSS®E

- Model Driven Transformation can be used for the conversion of power system network data between CIM in its RDF XML form into a PSS®E RAW text file
- This is converting between different formats (XML into whitespace delimited, column-oriented text form)
- It is also converting between different data models (CIM and the PSS®E data structures)
- The source and target data are thus very different

Open Grid Systems

PSS®E Data

```
0,    100.00, 31, 0, 1,  50.00 / PSS(R)E 31 RAW created by rawd31  SUN, APR 18 2010  10:53


     1,'NODE 1      ', 110.0000,1,   1,   1,   1,1.05000, -36.5869
     2,'NODE 2      ', 400.0000,1,   1,   1,   1,1.03808, -34.1462
     3,'NODE 3      ', 220.0000,1,   1,   1,   1,1.05241, -38.1991
     4,'NODE 4      ', 400.0000,1,   2,   1,   1,1.03401, -34.3306
     5,'NODE 5      ', 220.0000,1,   2,   1,   1,1.02963,  -3.5895
     6,'NODE 6      ',  15.7500,3,   2,   1,   1,1.04800,   0.0000
     7,'NODE 7      ',  10.5000,2,   1,   1,   1,1.04592, -34.2674
     8,'NODE 8      ', 220.0000,1,   2,   1,   1,1.02963,  -3.5895
     9,'NODE 9      ',  15.7500,2,   2,   1,   1,1.04700,  -1.8518
    10,'NODE 10     ',  21.0000,2,   1,   1,   1,1.04700, -19.6684
    11,'XAA_AB11    ', 400.0000,1,  99,  99,   1,1.03702, -34.2429
    12,'XAA_AB12    ', 400.0000,1,  99,  99,   1,1.03626, -34.2390
    13,'XAA_AB13    ', 400.0000,1,  99,  99,   1,1.03607, -34.2384
    14,'XAC_AD21    ', 220.0000,1,  99,  99,   1,0.99857, -16.6606
    15,'XAF_AK21    ', 220.0000,1,  99,  99,   1,0.99616, -16.6552
    16,'NODE 11     ', 220.0000,1,   1,   1,   1,1.01491, -29.9133
 0 / END OF BUS DATA, BEGIN LOAD DATA
     1,'1 ',1,   1,   1,   100.000,    90.000,     0.000,     0.000,     0.000,     0.000,   1
     3,'1 ',1,   1,   1,   200.000,    50.000,     0.000,     0.000,     0.000,     0.000,   1
     4,'1 ',1,   2,   1,   290.000,   280.000,     0.000,     0.000,     0.000,     0.000,   1
     4,'2 ',1,   2,   1,    10.000,    10.000,     0.000,     0.000,     0.000,     0.000,   1
     8,'1 ',1,   2,   1,   400.000,   250.000,    50.000,    25.000,    33.000,    43.000,   1
 0 / END OF LOAD DATA, BEGIN FIXED SHUNT DATA
 0 / END OF FIXED SHUNT DATA, BEGIN GENERATOR DATA
     9,'2 ',   150.000,    83.296,   200.000,     0.000,1.04700,     0,   250.000, 0.00000E+0, 2.30000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100
     9,'1 ',   140.000,    77.743,   200.000,     0.000,1.04700,     0,   250.000, 0.00000E+0, 2.30000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100
     6,'1 ',   597.443,   180.915,   600.000,     0.000,1.04800,     0,  1111.000, 0.00000E+0, 3.18000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100
     7,'1 ',    90.000,   100.256,   160.000,  -100.000,1.05000,     1,   300.000, 0.00000E+0, 2.00000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100
    10,'1 ',   118.170,    18.792,   200.000,  -200.000,1.04700,     0,   300.000, 0.00000E+0, 2.00000E-1, 0.00000E+0, 0.00000E+0,1.00000,1,  100
 0 / END OF GENERATOR DATA, BEGIN BRANCH DATA
     2,     13,'3 ', 1.45000E-4, 1.26500E-3,   0.04149,  900.00,  950.00,  850.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,  40.00,   1,1.0000
     2,     11,'1 ', 6.50000E-4, 7.50000E-3,   0.25120, 1000.00, 1050.00,  950.00,  0.00000,  0.00000,  0.00000,  0.00000,1,1,  30.00,   1,1.0000
```

Open Grid Systems

CIM RDF XML Data

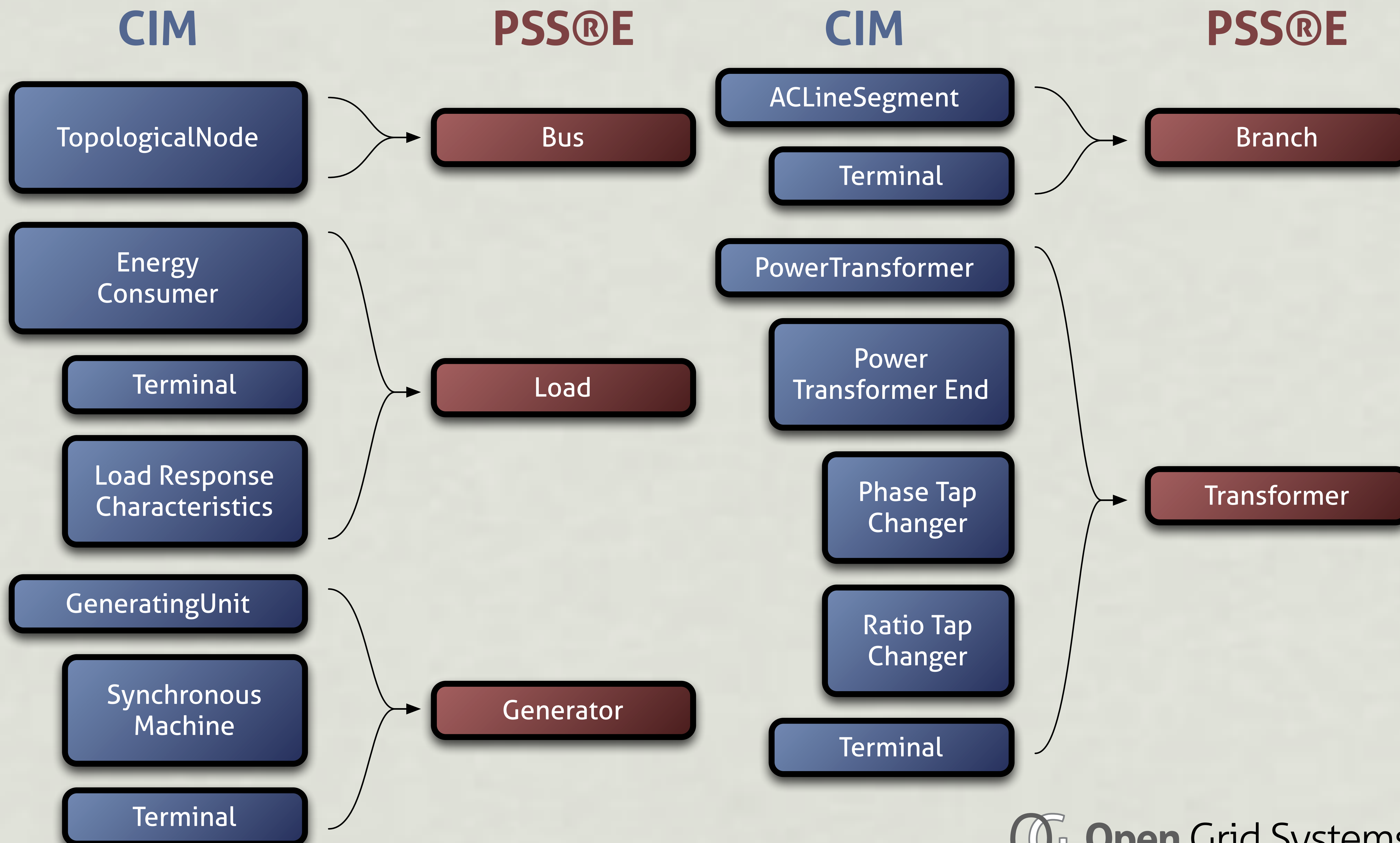

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:cim="http://iec.ch/TC57/2009/CIM-schema-cim14#"
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
 <cim:TopologicalNode rdf:ID="_2f048bf0-1b10-11e2-899c-005056c00008">
  <cim:IdentifiedObject.aliasName>1</cim:IdentifiedObject.aliasName>
  <cim:IdentifiedObject.name>NODE 1</cim:IdentifiedObject.name>
  <cim:TopologicalNode.TopologicalIsland rdf:resource="#_2f0464e0-1b10-11e2-899c-005056c00008"/>
  <cim:TopologicalNode.SvVoltage rdf:resource="#_2f048bf7-1b10-11e2-899c-005056c00008"/>
  <cim:TopologicalNode.BaseVoltage rdf:resource="#_2f048bf1-1b10-11e2-899c-005056c00008"/>
  <cim:TopologicalNode.ConnectivityNodeContainer rdf:resource="#_2f048bf4-1b10-11e2-899c-005056c00008"/>
 </cim:TopologicalNode>

 <cim:BaseVoltage rdf:ID="_2f048bf1-1b10-11e2-899c-005056c00008">
  <cim:IdentifiedObject.name>110.0KV</cim:IdentifiedObject.name>
  <cim:BaseVoltage.nominalVoltage>110</cim:BaseVoltage.nominalVoltage>
  <cim:BaseVoltage.isDC>false</cim:BaseVoltage.isDC>
 </cim:BaseVoltage>

 <cim:BusbarSection rdf:ID="_2f048bf2-1b10-11e2-899c-005056c00008">
  <cim:IdentifiedObject.name>NODE 1 B</cim:IdentifiedObject.name>
  <cim:Equipment.EquipmentContainer rdf:resource="#_2f048bf4-1b10-11e2-899c-005056c00008"/>
 </cim:BusbarSection>

<cim:EnergyConsumer rdf:ID="_2f059d67-1b10-11e2-899c-005056c00008">
  <cim:IdentifiedObject.name>1</cim:IdentifiedObject.name>
  <cim:Equipment.EquipmentContainer rdf:resource="#_2f048bf4-1b10-11e2-899c-005056c00008"/>
  <cim:EnergyConsumer.LoadResponse rdf:resource="#_2f059d69-1b10-11e2-899c-005056c00008"/>
 </cim:EnergyConsumer>

<cim:Terminal rdf:ID="_2f059d68-1b10-11e2-899c-005056c00008">
  <cim:IdentifiedObject.name>1 T1</cim:IdentifiedObject.name>
  <cim:Terminal.TopologicalNode rdf:resource="#_2f048bf0-1b10-11e2-899c-005056c00008"/>
  <cim:Terminal.ConductingEquipment rdf:resource="#_2f059d67-1b10-11e2-899c-005056c00008"/>
  <cim:Terminal.SvPowerFlow rdf:resource="#_2f059d6a-1b10-11e2-899c-005056c00008"/>
  <cim:Terminal.connected>true</cim:Terminal.connected>
  <cim:Terminal.sequenceNumber>1</cim:Terminal.sequenceNumber>
 </cim:Terminal>
```

M2M CIM to PSS®E

- Assuming the CIM model is taken directly from the UML the other components required are:
    - A **PSS®E information model** derived from its underlying structure
    - A **serializer** for PSS®E RAW files to write the data objects into PSS®E RAW format
    - A transform between the CIM classes and the PSS®E information model elements
- A **CIM RDF XML parser** must be written to convert the CIM RDF XML into **CIM data objects**

Open Grid Systems

Model Mapping



Open Grid Systems

Model Driven Transformation

- An example of a transformation using the **QVTO** language to map **CIM EnergyConsumer** to a **PSS®E Load** is shown below:

```
mapping CIM::IEC61970::Wires::EnergyConsumer:: getLoad() : PSSE::Load {
    -- Set the Bus to be the first (and only)
    -- Terminal's Topological Node mapped to a PSSE:Bus
    result.Bus := self.Terminals->first().TopologicalNode.map getBus();
    result.id := self.name;

    var p : Real := 0.0;
    var q : Real := 0.0;

    -- If the EnergyConsumer's Terminal has an SvPowerFlow associated
    -- with it then we use the values from it, otherwise we use the
      fixed
    -- real and reactive power values for the load
    if (self.Terminals->first().SvPowerFlow <> null) then{
        p := self.Terminals->first().SvPowerFlow.p;
        q := self.Terminals->first().SvPowerFlow.q;
    }else{
        if (self.isSet("pfixed")) then{ p := self.pfixed;}endif;
        if (self.isSet("qfixed")) then{ q := self.qfixed;}endif;
    }endif;
```

```
-- If a LoadResponse is set then this load will have Constant, Current
-- and Impedance components.  The PSS/E values are thus set based on the
-- total real/reactive power multiplied by the component
-- (which should add to 1.0)
if (self.LoadResponse <> null) then{
   if (not self.LoadResponse.exponentModel) then{
      Pmva := p * self.LoadResponse.pConstantPower;
      Qmva := q * self.LoadResponse.qConstantPower;
      Pcurrent := p * self.LoadResponse.pConstantCurrent;
      Qcurrent := q * self.LoadResponse.qConstantCurrent;
      Padmittance := p * self.LoadResponse.pConstantImpedance;
      Qadmittance := q * self.LoadResponse.qConstantImpedance;
   }else{
      -- PSS/E cannot deal with the exponent model so set all as MVA
      Pmva := p;
      Qmva := q;
   }endif;
}else{
   -- If no LoadResponse is present then only the constant power is set
   Pmva := p;
   Qmva := q;
}endif;

-- The PSS/E Area is mapped to the SubGeographicalRegion and the Zone to
-- the GeographicalRegion
result.Area := self.getSubGeographicalRegion().map getArea();
result.Zone := self.getGeographicalRegion().map getZone();
```

Model Driven Architectures

Benefits of MDA

Open Grid Systems

Model Driven Architectures

- Model Driven Transformation is one part of the Model Driven Architecture (**MDA**) approach to dealing with CIM data
- This offers a number of benefits:
  - By ensuring that all data has a **defined structure** with an information model it makes is easier for users to **understand** and **interpret** data independent of the serialisation format
  - **Model Driven Transformation** makes it easier to **write** and **maintain** the sharing of complex data between applications by focussing on the data structures rather than the serialisation formats

Open Grid Systems

Benefits of MDA

- By separating the parsing of the data from the transformation itself the process is **modularized**
- The parsing module is **re-usable** in a number of **processes** and simplify the transformations so they do not have to deal with the added complexity of file formats or database access.
- Other Model Driven technologies such as validation, *model to text* and **model-driven database** frameworks can use the same model artefacts

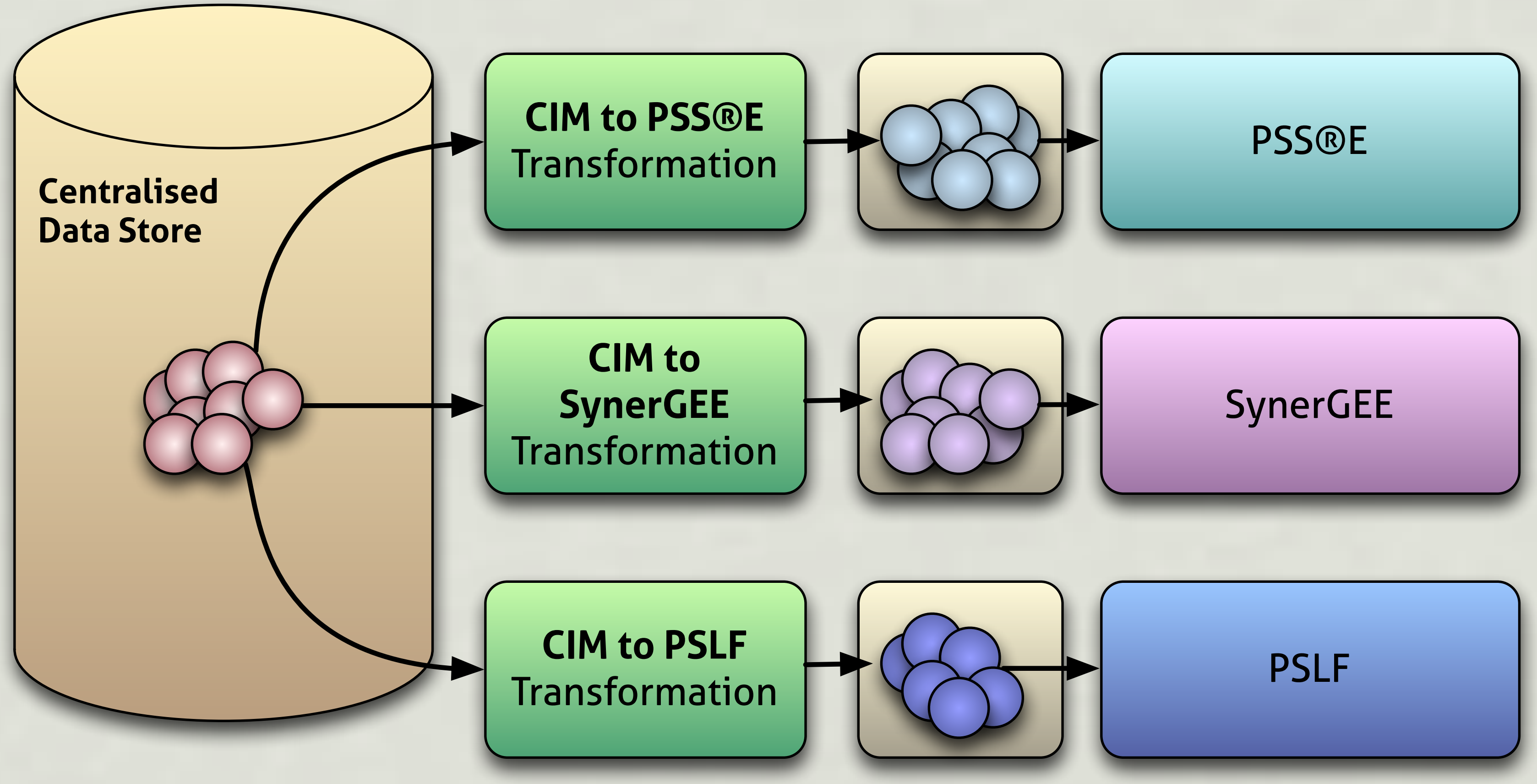
Open Grid Systems

Single Source

- A **single source** for network data requires a detailed model that covers the footprint of all applications
- This prevents **duplication** of **data** and **effort** in creating network models
- It **enables sharing** of models and results between applications from different vendors
- This often requires the conversion of the **common source** data into the formats of the target applications
- This single source must still be capable of providing data in a **format** that is supported by the analysis applications being used

Data Conversion

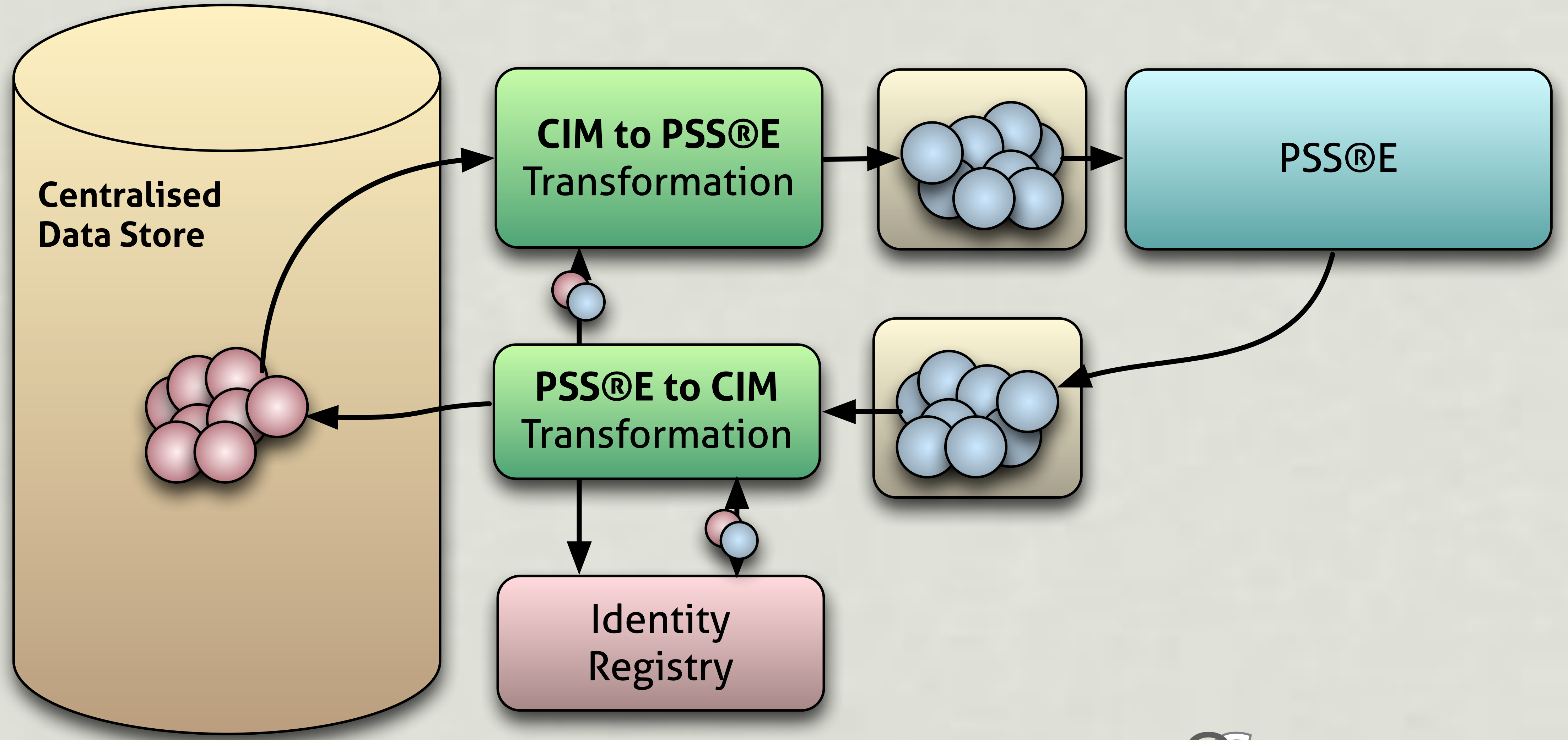
- For applications that do not support CIM directly transformations are needed to provide them with a model in their native format



Open Grid Systems

Stateful Transformation

- The exchange cannot be uni-directional as analysis results need to be integrated into the centralised store
- This requires **stateful transformation**



Open Grid Systems

Summary

Open Grid Systems

Model Driven Architectures

- A Model Driven Architectures (MDA) offers many benefits for **data management**
- Ensuring that all data is defined by a meta-model allows every system to **understand the structure** of the data and and **cross-system relationships**
- Exposing data to other systems in a **common structure** makes it easier to **integrate** the data from new and existing applications
- Tools and frameworks enable the serialisation interfaces to be **abstracted** away
- You now have **Enterprise Data** not Application Data

Open Grid Systems

Eclipse

- The Eclipse Platform provides a powerful set of frameworks for **implementing a Model Driven Architecture** and model-driven data management
- The Eclipse Modelling Framework is widely used by other Eclipse projects and provides a **mature modelling framework** and powerful **API**
- Model Driven Transformation using **QVTO** enables **structural data transformation** using an open **OMG** standard language
- This architecture can be used to facilitate **systems integration** and **application development**


Questions?
www.opengrid.com
alan@opengrid.com
susan@opengrid.com