phasor data open ECA platform

DESIGN DOCUMENT

June 30, 2016

DOE FOA 970
DE-OE-778

Document Purpose

This document describes the functional and non-functional requirements for:

1. The openECA platform
2. The included analytic systems that will:
   - Validate the operational readiness and performance of the openECA platform
   - Provide out-of-box value to those that implement the openECA platform with an initial collection of analytics

Design Document Contents

1. Introduction
2. An Example Use Case
3. Architectural Overview
4. Data Conditioning / Alarming
5. Data Integration Services
6. Common Analytics Interface (CAI)
7. Shared Platform Services
8. Electric System Model
9. Included Analytics Design
10. Appendices
11. References

DESIGN DOCUMENT

open ECA

1. INTRODUCTION

Project Partners

- Dominion Virginia Power
- Oklahoma Gas and Electric
- Southwest Power Pool
- Northwestern Energy
- Bonneville Power Administration
- Virginia Tech
- T&D Consulting Engineers
- Grid Protection Alliance
- DOE – Office of Electricity

openECA Project Summary

A better way to connect phasor data to analytics

Objective

To develop an open-source software platform that facilitates the development and production use of analytics that make use of high-fidelity synchrophasor data

2-Year Project Schedule

October 2015 – September 2017

- Final design – 6/30/16
- Alpha Version – 12/31/16
- Demonstration Begins - 6/30/17
- Version 1.0 released - 9/15/17

Project Status

Project Awarded
Sept. 2015

Value of Award

$ 5.0 M

Prime

Grid Protection Alliance

Analytics Development is Simplified

Today's Approach

- "Signal" paradigm
- Use C37.118
  - Socket management
  - Protocol parsing
  - Exception handling
- Local data buffering to support analytic cycle times
- Local configuration management

Using openECA

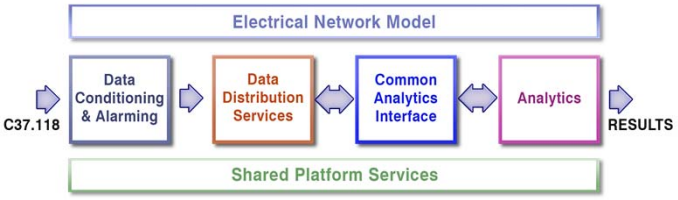
- Both standard and custom data objects
- An API (the CAI) that provides
  - Hi-performance pub/sub data access using standard messaging (e.g., Zero MQ)
  - Access to meta data services
  - Local data buffering options
- Starter templates provided
  - Matlab
  - C# / F# / VB (.NET)
  - C++
  - Java

Value to the Industry

- Lowers cost of addition of new production analytic tools
- Simplified end-to-end configuration and change management
- Improved availability of phasor data with greater visibility of phasor data quality
- Robust scalable solution to support phasor data infrastructure of any size
- Complements current phasor data architecture and supports integration with other data sources such as SCADA

Architectural Elements

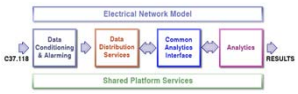
- Data Conditioning and Alarming*
- Data Integration Service
- Common Analytics Interface (CAI)
- Electric System Model*
- Shared Platform Services
- Analytics



* Optional to setup, i.e., not required to run openECA, but may be required for some analytics

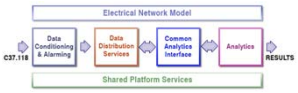
Data Service Summary

- Real-time and historical data acquisition
  - Adapter-driven data providers will include:
    - All common synchrophasor protocols
    - Common RDBM systems
    - OSI-PI and other historians
    - Other protocols, e.g., DNP3, Kafka, COMTRADE
- Device management
  - Automated connectivity
  - Data quality reporting
- Time-series data management



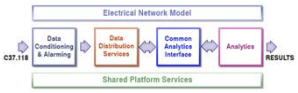
Common Analytics Interface Summary

- Server API (targets .NET)
  - Authorizes client data source connectivity
  - Provisions time-series data and metadata
- Client API (targets multiple platforms)
  - Manages server connectivity
  - Executes data filtering, organization and aggregation over user defined time-intervals
- Data Modeling Manager Tool
  - Defines data filtering, organization and aggregation
  - Trends incoming data sources



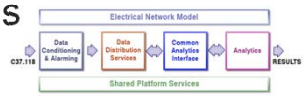
Shared Platform Services Summary

- Security and authentication services
- Time-series data transport with support for multiple data types
- Management of RDBMS connectivity, as needed
- Logging services



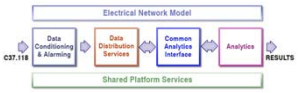
Data Conditioning and Alarming Summary

- Data conditioning will be provisioned through the Linear State Estimator (LSE):
  http://phasoranalytics.codeplex.com/
- The LSE will directly integrate with the Data Service for input and output
- Conditioned data will be made available to correlated groups along with associated real-time data where applicable
- Alarming data will be available based on signal-to-noise value deviations



Electric System Model Summary

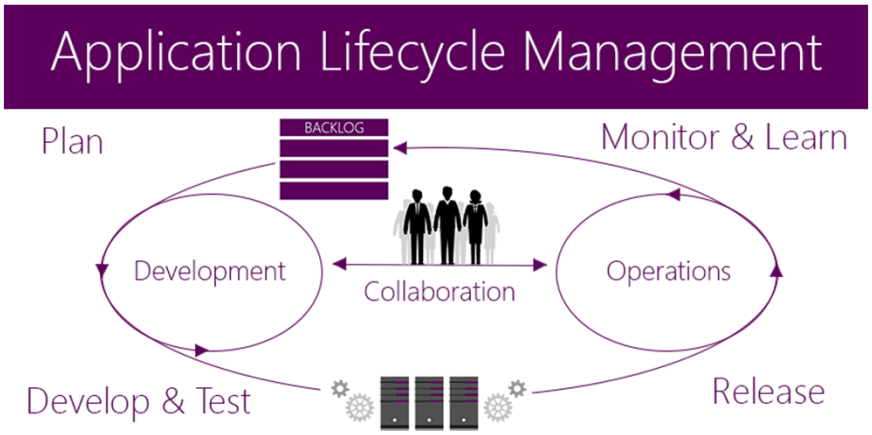
- LSE requires an electric system model that maps to time-series measurements that will be made available to analytics
- The model is defined as XML and will be provided to analytics upon demand to analytics that request the model
- Request and reception of XML based model will occur via CAI Client API



"Out-of-the-Box" Included Analytics

A. Localized Voltage-VAR Controller
B. PMU Instrument Transformer Calibration
C. PMU Synchroscope
D. Real-Time Impendence Calculator
E. Regional Voltage Control
F. Topology Estimator
G. Transmission Line Impedance Calibration
H. Oscillation Detection
I. Oscillation Mode Meter
J. Synchronous Machine Parameter Estimation
K. Acceleration Trend Relay Improvement

Development Principles



- https://msdn.microsoft.com/en-us/library/aa266497(v=vs.60).aspx

Development Approach

- Must build on existing production-ready open source software where possible, e.g.:
  - Grid Solutions Framework (GSF)
  - GSF Time-Series Library (TSL)
  - Math.net (www.mathdotnet.com)
- Must leverage proven open source technologies, e.g.:
  - ZeroMQ
  - log4net
  - .NET
  - Mono

Development Environment

- Visual Studio 2015 for primary code development and testing – for most components, primary target language will be C#
- GitHub will be used for source code versioning and continuous integration
- Web based UI components will target HTML 5 compatible browsers, e.g., newer versions of Google Chrome, Edge, or Internet Explorer 11

Testing / Code Acceptance

- Performance Testing
  - System will be stressed with combinations of high-volume and high-speed data transmission scenarios to make sure openECA will continue to perform well even if an analytic behaves poorly
  - Testing will ensure that openECA can be utilized with production-level data feeds without adverse affects to upstream systems
- Demonstration Testing
  - Included analytics will be deployed in appropriate environments at partner utility locations to verify openECA is functioning as expected

Product Releases

- Releases will be made available via GitHub:

https://github.com/GridProtectionAlliance/openECA/releases

https://help.github.com/articles/creating-releases/

- Project Schedule defines the following product releases:
  - Alpha – 12/31/2016
  - Beta – 4/30/2017
  - Version 1 – 9/30/2017 (but we'll really publish it about a month earlier)

DESIGN DOCUMENT

2. USE CASE

Creating a new analytic tool using openECA to provide phasor data

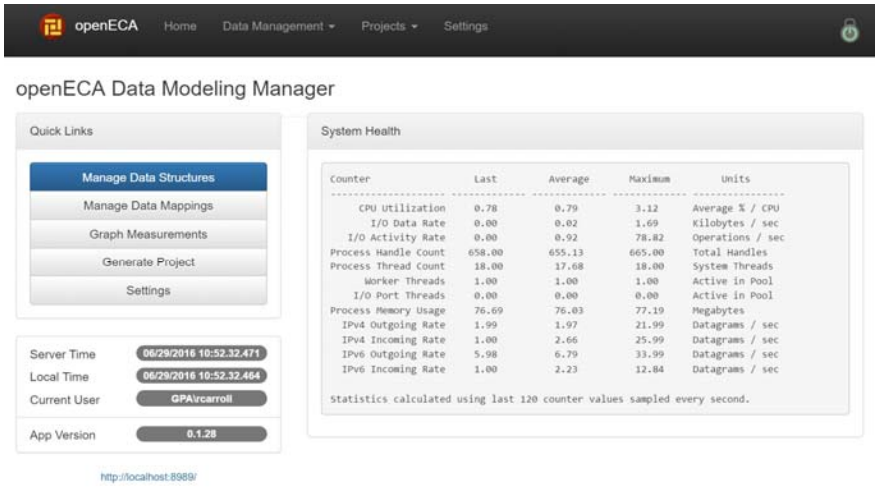
With openECA, a Paradigm Shift

- openECA defines a unified environment for modeling an analytic's:
  - Configuration
  - Data Structures and
  - Measurement Mapping
- openECA will use a common user interface to deploy analytics templates for most modern development tools and be used to help develop and debug an analytic without adversely affecting existing production data collection environments

Creating a New Analytic

- Development of a new analytic will begin with the openECA Data Modeling Manager Tool (provided along with the CAI Client API)
- The analytic developer will begin by using the tool to make a connection to one or more Server API instances and start reviewing the available data sources and associated measured values
- The developer will then select a target language for their analytic and use the tool to create a new project

The Data Modeling Manager Tool



Analytic Data Structure Definition

- The Data Modeling Manager Tool will allow the analytic developer to define two classes of data structures:
  - The first is the domain input called ***SourceData****
  - The second is the analytic product called ***ResultData****
- The contents of these data structures are under the complete control of the analytic developer
- The *SourceData* will be automatically populated with time-aligned incoming data over the desired data window, e.g., 1/30 of second or a two-second window
  - Auto-generated code that populates the *SourceData* structure will be available for the analytic writer to review and validate
- The *ResultData* will be populated by the analytic developer and returned at the end of processing

* Not a fixed name, developer will name these data structures

Analytic Entry Point

Either by using a template for a quick start or starting a new analytic from scratch, the analytic writer will be implementing their analysis from a single point of entry, i.e., a single function call into their analysis code called on an interval of their choosing:

ResultData Mapping SourceData Mapping

```
MyOutputType Execute(MyInputType input)
{
    // Analytic code goes here…
}
```

Analytic Results

- The *ResultData* structure will be used by the analytic developer to hold the calculated results of the analysis
- The analytic result data is defined as a data structure so that results:
  1) will flow back to the Server API as a unit and can easily disseminated to historians and other adapter based outputs
  2) be directly available as a complete input into other analytics

Analytic Development

- Since the analytic will be in the developer's desired target language, the actual tool used to develop and test the algorithm may be different for each analytic
- The Client API will be implemented as a component in the target development language and will do the heavy lifting for making connections to the Server API, subscribing to data, time-aligning the data and mapping the data to the SourceData structure
- The analytic development process will proceed with real-time data in the chosen language tool where the developer can write code, run, break, inspect and debug as the source tool allows

Analytic Debugging

- Most modern development tools allow some level of native ability to debug code under development
- The openECA Client API will be providing data in the target language and be directly usable in the development tools
- Where supported by the development tool, this will allow the developer to put break points in their code, run the application, and stop on a line of code to evaluate the current data set and intermediate calculations while not adversely affecting the source systems

Analytic Deployment

- Deployment of developed analytic will depend on language/platform used, however, the process usually involves copying compiled binary forms of the source code onto a host system, often using an installation script or application
- These deployments will include the analytic binaries along with Client API binaries for the target language
- Since it will be necessary to “map” analytic source data to different sources in different environments, deploying the openECA Analytic Modeling Tool along with the analytics will be important
- openECA will provide, as necessary and useful, installation applications / scripts to make sure the analytic modeling tool can be easily deployed into target environments along with the analytic binaries

Visualization Development

- Visualization applications can also benefit from the openECA platform
- Development of visualizations would proceed identically to those steps laid out for analytic development
- One difference would be that a visualization does not usually provide results, so in this case the ResultData structure for a visualization would simply be an empty structure (often null or void depending on language)*

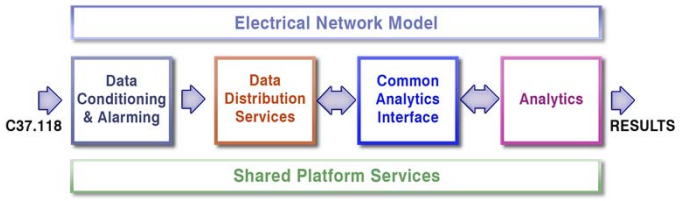
* Not all analytics will produce a result data set, data can be consumed and processed without providing feedback to the openECA system

DESIGN DOCUMENT

3. ARCHITECTURAL OVERVIEW

Architectural Elements

- Data Conditioning and Alarming*
- Data Integration Service
- Common Analytics Interface (CAI)
- Electric System Model*
- Shared Platform Services
- Analytics



* Optional to setup, i.e., not required to run openECA, but may be required for some analytics

Data Conditioning and Alarming Components

- *Linear State Estimator (LSE)* – defines the three-phase (or positive sequence) LSE core functionality needed for data conditioning
- *Phasor Signal Conditioner* – used to condition and smooth received data before LSE calculations execute*
- *Signal-to-Noise Ratio Calculator* – used to compute the signal-to-noise ratio of the phasor components (magnitude & angle) with best results during steady state conditions
- *Alarming Engine* – used to create new signals that will define alarmed states on detected abnormal data conditions

* No measurements are destroyed through conditioning processes and no information is lost. New signals and new information is created; users have the option of using unconditioned or conditioned data

Data Integration Service Components

- *Data Service* – manages incoming data sources, historical outputs and also hosts the common instance of the Server API
- *Data Service Manager* – manages the configuration, e.g., data sources and archives, of the openECA Service
- *Data Service Console* – provides low-level diagnostic management of the openECA Service
- *Performance Historian* – archives system performance statistics related to data sources and operational status
- *Data Historian* – archives time-series measurement data from configured data sources for use in analytics that needs access to historical data
- *RDMS Configuration* – persists system configuration information about the openECA system including device connection data and time-series measurement meta-data

Common Analytics Interface Components

- *Server API* – defines the base services for management and storage of analytic models, i.e., data structures and defined instances that include mapped measurements as well as management of measurement metadata
- *Client API* – defines the base services to manage analytic model configuration, measurement selection and bi-directional data transport
- *Data Modeling Manager Tool* – uses the CAI to manage:
  - Data structure definitions
  - Measurement mapping to identifiable data structure instances
  - Simple data trending and visualization

Shared Platform Services Components

- *System Security* – foundational layer provides security and authentication for client access to server-side data for the analytic user attempting to access the system
- *Integrated System Logging* – provides mechanisms for archiving operational state, system errors and general functional status messages to common locations, e.g., the Windows Event Log or a local log file
- *Time-series Data Transport* –provides a common set of measurement-centric data integration services that can process and exchange high-volume, high-speed time-series data using various data types

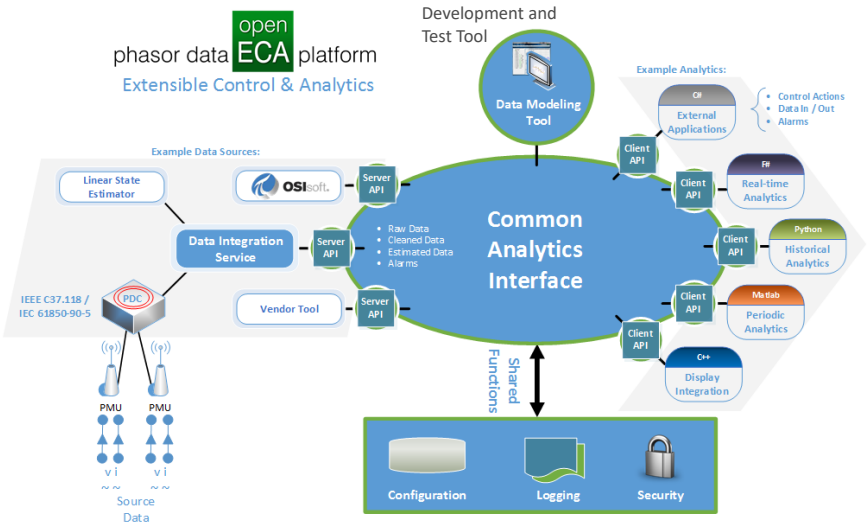
Electrical Network Model Components

- *XML Model File* – defines the XML elements that map electric system components to needed system measurements for the purposes of estimating system state – this includes modeling of branch impedances (including 3-phase), tap configuration, breaker status, phasor status word, substation and node data, phasor data, circuit breakers, switch data, two-winding transformers and current flow groups
- *Model Tools* – applications and libraries for building, maintaining and troubleshooting a full breaker/switch network model in both three phase or positive sequence representation as well as managing the system matrix representations of that network

openECA Provided Analytics

A. Localized Voltage-VAR Controller
B. PMU Instrument Transformer Calibration
C. PMU Synchroscope
D. Real-Time Impendence Calculator
E. Regional Voltage Control
F. Topology Estimator
G. Transmission Line Impedance Calibration
H. Oscillation Detection
I. Oscillation Mode Meter
J. Synchronous Machine Parameter Estimation
K. Acceleration Trend Relay Improvement

System Data Flow Diagram



Development Challenge Summary

- *Managing Feature vs Component Tension* – feature focused development provides good end-to-end functionality cohesion but usually lacks the code reusability that comes from a more component based focus
- *Business Value Development* –a strong emphasis on delivering business value is a proven advantage of software development, however, understanding of what constitutes *business value* can often be a hurdle
- *Development for Agility vs Stability*– a focus on extensibility will provide a much more useful platform for data and systems integration but comes at the cost of complexity during development – early on in development cycles, more complex code is typically less stable

DESIGN DOCUMENT

open ECA

4. DATA CONDITIONING / ALARMING

Data Conditioning and Alarming Components

- Linear State Estimator (LSE)
- Phasor Signal Conditioner
- Signal-to-Noise Ratio Calculator
- Alarming Engine

Linear State Estimator

- The Linear State Estimator (LSE) was developed at Virginia Tech and deployed at Dominion Virginia Power as part of the DOE Technology Demonstration Project 'Three Phase Linear State Estimator and Its Applications'
- The LSE is configurable as either a positive sequence LSE or a 3P LSE and is designed to provide model-based data conditioning and increased network observability
- The current format of the LSE is as a C# (.NET) library that interfaces with the Grid Solutions Framework Time-Series Library environment (e.g., openPDC through an Action Adapter)
- The LSE software-implementation is maintained as open source at:

  http://phasoranalytics.codeplex.com
- The LSE includes the core libraries, necessary GSF adapters, network model editor, an offline-version of the LSE for troubleshooting models and measurements, and additional real-time synchrophasor data-conditioning algorithms

Phasor Signal Conditioner

- The LSE code base also includes algorithms for filtering, smoothing, and replacing missing data from a phasor stream in real-time
- These algorithms were developed in a previous project by Virginia Tech and have been tested against Dominion synchrophasor data
- The Phasor Signal Conditioner can provide a conditioned stream of phasors to certain classes of analytics where macro-improvements to signal quality overshadow a small loss of information due to smoothing
- The LSE can accept the conditioned phasor data from this algorithm. However, this is not a requirement; the LSE can also accept raw phasor data as well
- The Phasor Signal Conditioner takes one phasor as input and outputs several phasor values and several data conditioning metrics that contain information about the output measurements

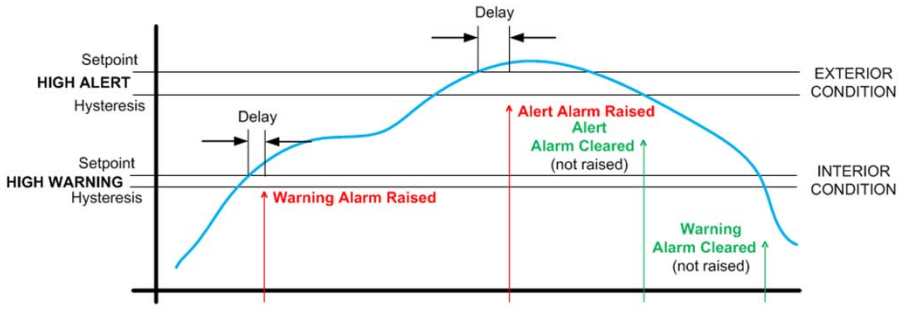
Signal-to-Noise Ratio Calculator

- The Signal-to-Noise (SNR) Ratio Calculator is a simple adapter included in the LSE code base that is used for measuring data quality
- The SNR calculator does not create modified phasor values but rather calculates an approximation of the SNR of the components of the phasor signal (under the assumption of quasi-steady state conditions, the phasor components approximate DC signals)
- Monitoring of this 'SNR' can provide metrics on data quality that can be alarmed on
- Additionally, this kind of metric can also provide insight into equipment health

Alarming Engine

- openECA will provide an alarming engine within the Data Service that will allow creation of new signals to alarm on abnormal data conditions
- System will allow groups of alarms to be created for any given source "measurement point", e.g.:
  - Phasor magnitudes or angles
  - Calculated values, for example:
    - Difference between a measured value and a value from the state estimator
    - Rate of change of a measured value
- Engine will make alarming results available to analytics as new measurements

Alarm Configuration Example

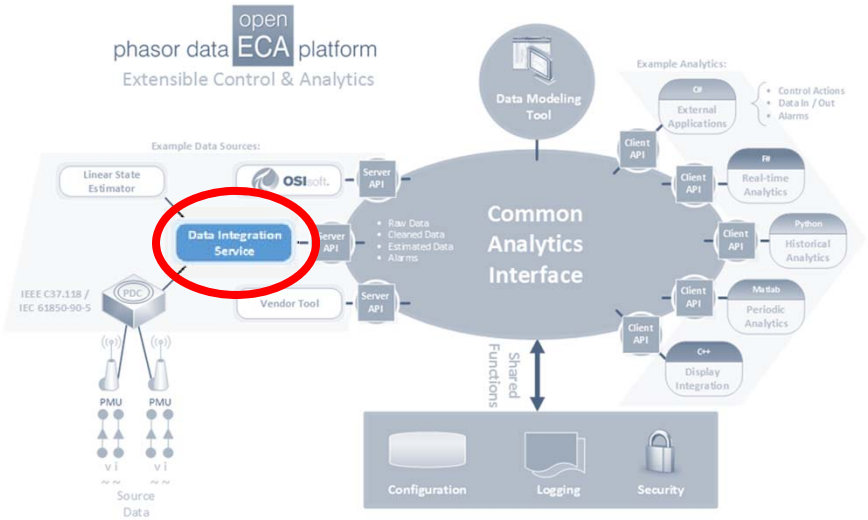


Warning and Alert Alarms are raised at set points and are cleared only after falling below a specified hysteresis.

DESIGN DOCUMENT

5. DATA INTEGRATION SERVICES

Data Integration Services



Data Service Components

- Data Service
- Data Service Manager
- Data Service Console
- Performance Historian
- Data Historian
- RDBMS Configuration

Data Service

- Extensible adapter based architecture will allow integration of multiple data sources
  - Real-time (e.g., PMU, SCADA, State Estimation, Analytic Results)
  - Historical (e.g., OSI-PI Historian, local archives, SQL databases)
  - Semi-static (e.g., COMTRADE or PQDIF files) in looped or on-demand publication
- Maps data sources to time-series values
  - Support for common fundamental data types (e.g., float, int, bool, string)
- Common host for openECA Server API

Data Service Functions

- Measurement Definition
  - Flexible Data Types
  - Automatic Creation from Phasor Sources
- Measurement Validation
  - Data Quality Testing
  - Flat-line Detection
- Data Acquisition and Routing
- Adapter Configuration Management
- Linear State Estimation Value Integration
  - Available via secondary install and configuration
  - Allows grouping of actual and estimated values

Data Service Manager

- The Data Service Manager will be designed as a desktop client for configuration and monitoring of the Data Service
- The Manager will be used to make connections to remote systems, e.g., Phasor Data Sources and SCADA systems as well as configure system parameters*
- The Data Service will require an RDBMS system for storage of system configuration – the manager tool will be used to manage this configuration

* Note that it is the Data Server *service*, not the *manager*, that will actually make connections to devices for streaming time-series data, e.g., synchrophasor data – the manager is the UI tool

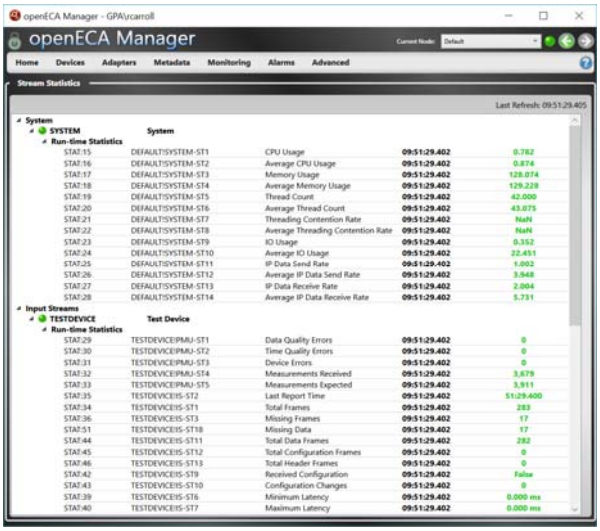
Data Service Console

- The Data Service Console application is a diagnostic tool used for monitoring the Data Service for current system activities as well as issuing low level system commands as may be needed during production operation
- The Console application will be typically focused towards IT professionals that will use the tool to quickly ascertain status of key system adapters

Performance Historian

- An instance of the openHistorian will automatically deployed with the Data Service for archival of device and connection statistics for ongoing analysis, troubleshooting and auditing
- Multiple statistics will be gathered every ten seconds from all input, action and output sources

Performance Historian Example Metrics



Data Historian

- An instance of the openHistorian will made available so it can be optionally deployed with the Data Service for archival of data from incoming sources and archival of analytic results if desired
- Data can be stored at full resolution and will be locally compressed with a high-speed lossless compression algorithm

RDBMS Configuration

- A relational database management system will be required to store configuration information related to the Data Service
- The openECA Data Service will at least support the following common RDBM systems:
  - MS SQL Server
  - Oracle
  - MySQL
  - PostgreSQL
  - SQLite (no extra DB installation required)

Data Integration Service: Non-functional Requirements

- Deployed system components must be highly available and be able to run for extended periods, e.g., months, without any required user intervention
- Must support all of the most common RDBMS implementations, e.g., SQL Server, Oracle, and MySQL, for flexible configuration management
- Design must be extensible, e.g., allowing use of adapters to add features without a need to reinstall application
- Fault tolerant deployment options must be supported, e.g., fail-over cluster and/or load balanced clusters
- Analytics written in different target languages or deployed on differing platforms must still be interoperable
- Code must be deployed as open source using permissive licensing and only use compatible open source tools that support permissive licenses
- Must support horizontal scaling through data partitioning and distribution across multiple systems
- Security must be implemented throughout the system by design

DESIGN DOCUMENT

6. COMMON ANALYTICS INTERFACE

Common Analytics Interface Components

- Server API
- Client API
- Data Modeling Manager Tool
  - Will create analytic projects that will contain test harness application, production deployment service and needed installation scripts

CAI Server API

- Configuration Serialization
  - Data class structure definitions
  - Labeling and identification of class instances with full measurement mapping
- Security Management
  - Validating clients and connections
  - Validating access to needed measurements
- Open API for Multivendor Support
  - Server API will define minimum requirements needed to implement a server side solution that will allow vendors to support analytics written using the openECA Client API

Target Platforms for CAI Server API

- Windows
  - Version 7 or greater / 64-bit only
- Linux
  - Using Mono version 4 or greater
  - Will target latest Ubuntu for testing
- Mac OSX
  - Using Mono version 4 or greater

All options may not be available on all target platforms

Configuration Serialization

- Server will proxy all measurement meta-data provided by Data Service to Client API instances
- Server API will serialize identifiable data structures that are defined by the analytic developer using the Developer and Visualization tool – these data structures become the “inputs” and “outputs” of the analytics
  - User data structures will include mappings to measurements as defined in meta-data that is maintained by the Data Service

Security Management

- Using the security and authentication tools provided in the Shared Platform Services, the Sever API will authenticate connections from Client API instances
- The Server API will also validate access to data sets (i.e., groups of defined measurements) based on identity of authenticated user of Client API

CAI Client API

- User Defined Data Structure Definitions
  - Automatically referenced within analytic tools, the Client API will exist as a set of base services used to retrieve sets of user defined data for input and outputs
- User Defined Data Collection Windows
  - The Client API will handle time-alignment of incoming data and support creation of "windows" of data to an analytic, e.g., providing a one second window of data to an analytic every second

User Defined Data Structure Definitions

- Analytic developers will use the Data Modeling Manager tool to create custom collections of measurements that will form the data structures that will become the input and output of openECA
- A defined data structure will hold an instance of synchronized data all collected at a given time interval
- Custom data structures will contain:
  - Individual fields mapped to measurements, or
  - Arrays of fields with common identity, i.e., an array of measurements of the same type, e.g., all available synchrophasor frequencies

Short Term Analysis: Volatile Data Windows

- Although a user defined data structure only defines a collection of measurements for a single time instance, the CAI Client API will allow analytic developers to define windows of data, e.g., a full second of data, before calling the analytic function
- The CAI Client API will manage and call the analytic function with the data that has been collected on the user specified data window
- These collection windows will be stored in volatile memory and are a good option when the total data volume storage is reasonable for memory storage over the desired window

Long Term Analysis: Persisted Data Windows

- For longer data collection windows, the CAI Client API will persist collected data to disk, local to the machine running the analytic*
- Once the data collection window has expired, the analytic function will be triggered and allow the analytic function to read the collected data

* Internally this will just use a light-weight instance of the openHistorian and needed APIs for local persisted storage of structure data

Production Level Target Platforms

- The following platforms will be supported for analytic algorithm development and are considered applicable for production level deployments:
  - .NET (C# / F# / VB)
  - C++
  - Java

Test Level Target Platforms

- As time permits, the following platforms will be added for analytic algorithm development but will only be considered applicable for algorithm validation and testing:
  - MATLAB
  - Python
  - JavaScript
  - MS Excel

Integrating 3rd Party Analysis Tools

- Depending on target platform, it will often be useful to use existing libraries to speed analytic development – for more production platform deployments, projects templates will offer references to common handy third party tools, e.g.:
  - .NET Platforms:
    - Math.NET
    - Deedle
  - C++
    - Eigen

Data Modeling Manager Tool

- Allows definition of logical groupings of measured values that represent organized structures of data, i.e., *data structures* – these definitions will map directly to language specific equivalents, e.g., a C language struct, an F# type or a class in C++, C# or Python
- Creates new uniquely labeled *identifiable instances* of data structures that directly map time-aligned measurement values to the structure fields – this will be auto-generated code, in the target language, that handles measurement-to-structure field mapping
- Using a real-time connection to the Server API, provides a *visual representation* of user defined data structures with updating values and simple trending with easy measurement lookup tools for mapping measurements to structure fields

Target Platforms for Data Modeling Manager tool

- Windows
  - Version 7 or greater
- Linux
  - Using Mono version 4 or greater
  - Will target latest Ubuntu for testing
- Mac OSX
  - Using Mono version 4 or greater

All options may not be available on all target platforms

DESIGN DOCUMENT

open ECA

7. SHARED PLATFORM SERVICES

Shared Platform Services Components

- System Security
- Integrated System Logging
- Time-series Data Transport

System Security

- Using common authentication options available on the target platform, the Server API will authenticate connections being established from Client API instances – this authentication will be for the user requesting analytic data
- Data transport will be secured using elliptic curve encryption at the ZeroMQ transport layer, see:

  http://curvezmq.org/page:read-the-docs#toc1

Integrated System Logging

- For .NET application platforms, tools, e.g., log4net*, will be available to use for common logging functionality in the APIs
  - APIs will allow custom logging application to log files and platform specific logging options, e.g., the Windows Event Log or syslog
- For other platforms, an appropriate logging option will be implemented, e.g., syslog or text file – as recommended by the platform and analytic host environment

* This is an available tool that will provided to the APIs, analytic developers can opt to use this and/or do their own custom logging

Time-series Data Transport

- Serialization
  - Advanced Lossless Compression
  - 3rd Generation Time-series Data Exchange, i.e., Gateway Exchange Protocol version 3 (GEP3) that supports various data types
- Transport
  - Will use ZeroMQ as a transport layer for widest target platform integration and speed

Supported Data Types for Measurements

- The time-series *data transport* will support various native data types for individual measured values, including*:
  - Floating-point (32 and 64-bit)
  - Integer (various sizes, 8, 16, 32 and 64-bit for signed and unsigned options)
  - Byte arrays (up to 64 bytes)
  - Short strings (up to 32 Unicode characters or 64 ASCII bytes)

* This is list is not exhaustive – however, the list will be intentionally limited to simplify *transport* serialization options – more complex types are provided via analytic developer custom type definitions

DESIGN DOCUMENT

open ECA

8. ELECTRIC SYSTEM MODEL

Data Model Components

- XML Model File
- LSE Model Tools

XML Model File

- LSE Model
  - The LSE model is defined in the source code of the LSE system
  - The actual model file is a serialized copy of the Network object from the LSE system that can be used not only to instantiate a network model but also to serialize and de-serialize network snapshots with measurement data
  - The serialized model will be made to openECA client analytics on-demand
- CIM Model
  - Where applicable in environments where a CIM model is being maintained and available, this model will also made available to clients

XML Model File Example

```xml
<?xml version="1.0" encoding="utf-8"?>
<Network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Model PhaseConfiguration="ThreePhase">
      <Companies>
         <Divisions>
            <Substations />
            <TransmissionLines />
         </Divisions>
      </Companies>
      <VoltageLevels />
      <TapConfigurations />
      <BreakerStatuses />
      <StatusWords />
   </Model>
</Network>
```


LSE Model Tools

- Network Model Builder – tool for developing a network model for the LSE
- Offline LSE – tool for running the LSE in offline mode to troubleshoot and validate the model being developed
- Measurement Sampler – adapter that is used to provide snapshots of a frame of measurements provided to the adapter:
  - Subscribes to a fixed set of measurements (often all measurements of a given type such as all voltage and current phasors or all frequency measurements)
  - Through user-issued commands, the sampler will serialize the current frame of measurements into an XML file
  - The serialized XML file is used to support troubleshooting activities, particularly for the offline LSE troubleshooting tool
- Online Documentation:
  - http://phasoranalytics.codeplex.com/wikipage?title=Modeling%20Branch%20Impedances

DESIGN DOCUMENT

9. ANALYTICS DESIGN

openECA Provided Analytics

A. Localized Voltage-VAR Controller
B. PMU Instrument Transformer Calibration
C. PMU Synchroscope
D. Real-Time Impendence Calculator
E. Regional Voltage Control
F. Topology Estimator
G. Transmission Line Impedance Calibration
H. Oscillation Detection
I. Oscillation Mode Meter
J. Synchronous Machine Parameter Estimation
K. Acceleration Trend Relay Improvement

Four Use Types

- Real-Time Observation (Real-Time)
- Real-Time Control
- Off-Line, Periodic
- Off-Line, Ad Hoc Use (Off-Line)

Three Use-Readiness Expectations

- Research-Grade (Alpha)
- Prototype-Grade (Beta)
- Production-Grade

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time Control
Prototype-Grade

A. Localized Voltage-VAR Controller

VirginiaTech Dominion

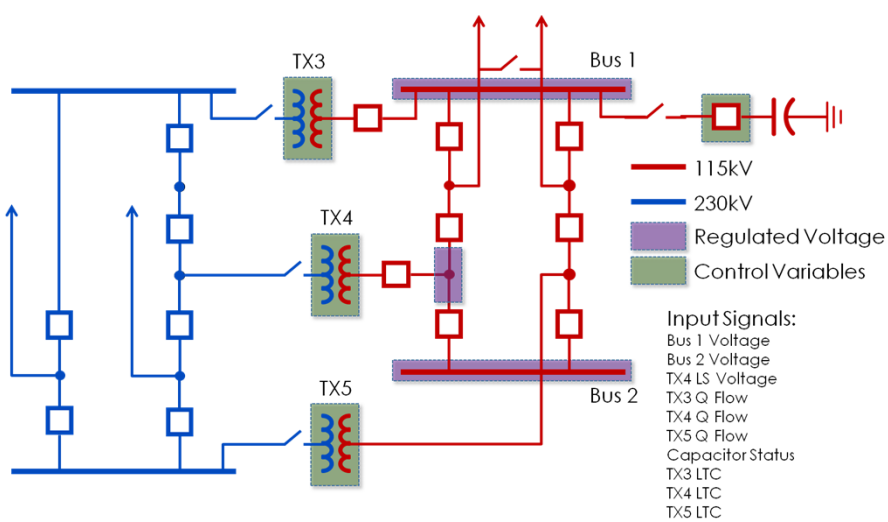
Purpose of the Analytic

- The purpose of the **Localized Voltage-VAR Controller** analytic is to:
  - Demonstrate end-to-end, synchrophasor driven control using an existing, proven, controller design.
  - And through this process, attempt to answer basic questions about synchrophasor-based control.
  - This will be accomplished by:
    - Porting existing, isolated control schemes for capacitor, reactor, and transformer tap controls from Dominion's EMS where the signal inputs for those control schemes overlap with locations with synchrophasor measurements

Analytic Algorithms and Methodology

- The analytic will utilize ***existing*** controllers from the Dominion EMS – simple controllers that use signals of voltage, real and reactive power to control transformer taps and coordinate with other voltage controlling devices such as capacitors, reactors and SVCs.
  - 43 existing SCADA-EMS based schemes across the transmission network.
  - 30/43 sites already have PMU/PDC installed
- The solution will use a generic, template-able controller that can be configured to achieve the functionality for each existing, individual controller.

Analytic Algorithms and Methodology



Functional Requirements

- The analytic will consist of a single on-line adapter as a generic controller that can be customized through a file-based configuration to enable the desired functionality. Therefore, this controller will not be specific to the Dominion system. The configurations for the Dominion system can serve as examples.
- The controller configurations should be in an XML file format.
- The analytic should be developed in C# using the latest version of .NET that is compatible with the openECA and the GSF.
- The adapter should utilize the 'Settings' dictionary mechanism to configure the adapter as much as possible. The analytic should de-serialize a configuration file for additional settings that cannot be succinctly or clearly expressed as adapter 'Settings'.
- The adapter should accept as input, (1) positive sequence phasors (V & I) as specified in the configuration (2) any Status Word and data quality information available for the aforementioned positive sequence phasors and (3) any topology related signals such as breaker statuses as specified in the configuration.

openECA Provided Requirements

- The **Localized Voltage-VAR Controller** analytic requires the following functionality from the openECA platform.
  - Requires time aligned measured values per adapter configuration
  - Requires an output/calculated value delivery mechanism
  - Requires a control signal delivery mechanism
  - Requires the use of adapter settings for configuration
  - Requires the ability to send commands to the adapter
  - Requires the ability to receive status updates from the adapter

External Functional Requirements

- The analytic requires controller designs ported from the EMS voltage/VAR controllers.
- The analytic will require many file-based configurations (one for each controller).

Expected Computational Resources

- RAM (Low)
- CPU (Low)
- File System (Low)

Testing and Demonstration

- Initial testing and development can utilize archived SCADA data for the signals that normally drive the control functions in the EMS. Success at this stage is characterized by reproducing the control signals created by the EMS under the same conditions.
- Next, testing can be done utilizing the RTDS and Dominion network models and synthesized synchrophasor data streamed to a test instance of openECA. Success at this stage is characterized by keeping the target voltage within the given range while eliminating control signal modulation when the controller is moving between states. The control signal should be fed back into the RTDS simulation to control the desired equipment.

Testing and Demonstration

- Finally, the analytic should be tested in the openECA platform using real-time synchrophasor data from the Dominion system. The control signal may either be blocked (no real control action) or fed back into our SCADA system to deliver a real control action. Success at this stage depends on the demonstration methodology. For a blocked signal, the control signal should nearly match the control signal generated by the EMS for the same controller. For an actually SCADA delivered control signal, success will mirror that of the RTDS testing.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Off-Line, Periodic
Prototype-Grade

B. PMU Instrument Transformer Calibration

VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **PMU Instrument Transformer Calibration** analytic is:
  - To reduce the impact of complex bias errors caused by current and potential transformers (as allowed by IEEE C57.13 and IEEE 57.16.6) that result in errors in synchrophasor measurements.
  - To form a basis for the off-line **Transmission Line Impedance Calibration** analytic.
  - This is accomplished by estimating the ratio correction factors and phase angle correction factors (referred to in this document simply as *ratio errors*) of the instrument transformers .

Analytic Algorithm and Methodology

- The analytic should compute estimated ratio errors for instrument transformers that provided metering for PMUs in the Dominion network. This is subject to:
  - The presence of at least one high quality voltage phasor measurement from a revenue quality PT (i.e. the PT design should minimize RCF and PACF so that the resultant phasor can be assumed to be a perfect measurement)
  - The ability to form a tree of measurements connecting all metering points where the ratio errors should be estimated.
  - The availability of periodic samples of synchrophasor data during a 24 hour period for all metering points in the tree of measurements such that many different loading conditions are included in the sample set.

Analytic Algorithm and Methodology

- The measured voltages, V_{im}, and currents, I_{im}, are related to the actual quantities by:

$$V_{im} = Rv_i V_i \qquad I_{im} = Ri_i I_i$$

where Ri_i and Rv_i are unknown, complex, constant numbers.

- The reciprocal of the ratio errors, Kv_i, and Ki_i, are the ratio correction factors:

$$V_i = Kv_i V_{im}$$

Analytic Algorithm and Methodology

- The fact that the ratio factors multiply the unknown voltages and currents combined with the existence of PMU errors suggests the methodology:
  - Take a sufficiently large number of measurements where the actual voltages and currents change between measurements sets while the ratio errors do not
  - Average to reduce the PMU effects of the PMU errors
  - Use at least one high accuracy voltage phasor measurement

Analytic Algorithm and Methodology

- If I_m is a $2 \, x \, m$ array of the two currents for m times points and V_m is a $2 \, x \, m$ array of the two voltages for the same m times points, then $\hat{Z}$ is the least squares solution of:

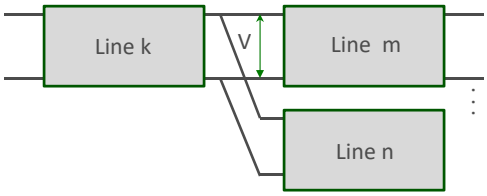
$$\hat{Z} * I_m = V_m \quad \hat{Z} = \begin{bmatrix} Z_{11}\dfrac{Ki_1}{Kv_1} & Z_{12}\dfrac{Ki_2}{Kv_1} \\ Z_{21}\dfrac{Ki_1}{Kv_2} & Z_{22}\dfrac{Ki_2}{Kv_1} \end{bmatrix}$$

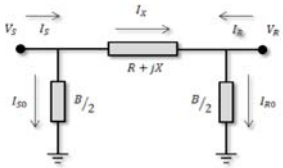
Analytic Algorithm and Methodology

- If the line impedances are known, then the ratios of correction factors can be obtained for each line from the archived data (that includes a single '*perfect*' voltage measurement).
- It is important that the m time points involve significant changes in the measurements to avoid ill conditioned equations

Analytic Algorithm and Methodology

- The accurate voltage measurement is propagated to the next π-section in the tree because the bus voltage at the end of one line is the input voltage to one or more new π-sections further down the tree.
- The current injections are not required in the methodology for this analytic.





Functional Requirements

- The analytic should consist of two components: (1) an on-line adapter and a (2) off-line application.
- The on-line adapter should assess each frame of the streaming synchrophasor data and generate flags to indicate (1) which frames meet the data quality and network topology requirements of the analytic's methodology and (2) to what degree those requirements are met.
  - The adapter should be developed in C# using the latest version of .NET support by openECA and the Grid Solutions Framework.
  - The adapter should utilize the 'Settings' dictionary mechanism to configure the adapter as much as possible. The analytic should de-serialize a configuration file for additional settings that cannot be succinctly or clearly expressed as adapter 'Settings'.
  - The adapter should accept as input, (1) positive sequence phasors (V & I) from all PMUs that lie on the largest tree of metering points (i.e. a subset of all phasor measurements) and (2) any Status Word and data quality information available for the aforementioned positive sequence phasors and (3) any topology related signals such as breaker statuses.
  - The adapter should generate integer-based output flags that should be published to the openECA platform and archived along side all PMU data.
  - The criteria that the adapter uses to flag frames of synchrophasor data should be configurable.

Functional Requirements

- The offline application should encapsulate all functionality of the algorithm.
  - The application must be able to estimate the ratio errors of the instrument transformers driving PMUs along the tree of measurements as described by the methodology.
  - The application must be able to take into consideration constraints imposed by network topology changes and data loss during the time frame of the sample data set.
- The application must be able to determine which timestamps in the archive correspond to data flagged as valid by the adapter.
- The application must be able to (1) retrieve all data flagged by the adapter as valid or (2) provide a suitably formatted query string to enable manual retrieval of all data flagged as valid by the adapter.

Functional Requirements

- The application must have functionality to intelligently select sample measurement sets from the data retrieved from the archive as well as to enable human selection or overriding of computer chosen data sets.
- The application must provide output results in a to-be-determined, standardized, XML format that will be compatible with the Linear State Estimator and the **Transmission Line Impedance Calibration** analytic.
- The computational component of the application must be developed in C# using the latest version of .NET compatible with the openECA and GSF and should be compiled as stand-alone libraries. The computational component of the application should be designed with reusability in mind where the is opportunity for overlap with the **Transmission Line Impedance Calibration** analytic.
- The application should have a graphical user interface (GUI) to enable ease of use by non-SME personnel. The GUI should be developed using XAML and C# using the MVVM design pattern. The GUI should be designed with reusability in mind where there is opportunity for overlap with the **Transmission Line Impedance Calibration** analytic.

openECA Provided Requirements

- The **PMU Instrument Transformer Calibration** analytic requires the following functionality from the openECA platform.
- The on-line adapter requires:
  - Requires time aligned measured values per adapter configuration
  - Requires an output/calculated value delivery mechanism
  - Requires the use of adapter settings for configuration
  - Requires the ability to send commands to the adapter
  - Requires the ability to receive status updates from the adapter
- The off-line application requires:
  - The archive of all values provided to and generated by the openECA platform in a local GSF-based historian
  - A mechanism for the application to access specific values from the archive asynchronously on a regular basis.

External Functional Requirements

- The off-line application may utilize MATLAB or Python scripts to perform some types of data manipulation.
- The off-line application and the on-line adapter will utilize the same network model (and measurement mapping) that is used by the Linear State Estimator.
- Minor modifications must be made to the Linear State Estimator code base to support this analytic, however, this analytic is not to be directly integrated with the LSE or the LSE code base.
- The LSE code base will be a dependency of both components of this analytic to take advantage of the previously developed network model management and topology processing.

Expected Computational Resources

- RAM (Low)
- CPU (Medium in real-time, Medium off-line)
- File System (High for large systems and archived data)

Testing and Demonstration

- ***Note:*** PMU data already has unknown ratio errors so validation of the algorithm using archived data alone is difficult. Therefore, initial steps will utilized models or synthesized data with known ratio errors.

- Initial testing and development will be accomplished by using PSSE models to synthesize data sets. These data sets will subsequently have known ratio errors introduced to make the data realistic. Success during this iteration is characterized by the ability of the analytic to determine the experimentally induced ratio errors. This stage will not require usage of the openECA platform. PMU data loss will be considered at this stage.

Testing and Demonstration

- Subsequent testing will be accomplished by utilizing Dominion's RTDS with a realistic model of the Dominion system that includes instrument transformers to generate streaming synchrophasor data. The data will be generated over a time window sufficient enough to emulate the 24-hour load curve of the Dominion region (but may not necessarily be 24 hours in length). The data will be archived and then utilized by the analytic to demonstrate the ability to estimate the ratio correction factors of the instrument transformers. PMU data loss will be simulated to demonstrate the robustness of the algorithm. Success during this iteration is characterized by the ability of the analytic to determine the ratio correction factors of the instrument transformers in the RTDS model.

Testing and Demonstration

- An alternate option for testing and demonstration is to use archived currents (assumed to be perfect) to produce voltages from known line models and then add known ratio errors and PMU errors to both current and voltage to check algorithm performance.

- The final test and demonstration of the analytic will be to utilize the openECA platform to archive Dominion's synchrophasor data and to run the analytic once for every 24 hours of data collected (note that the input data set for the analytic is only a subset of the 24 hours of data). Success in this final iteration is characterized by (1) results that have consistent and realistic values of ratio errors and (2) a robustness to both changes in network topology and data loss.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time
Production-Grade

C. PMU Synchroscope

VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **PMU Synchroscope** is to provide synchroscope functionality to a remote location (i.e. the control room) through:
  - Manual (computer supervised) controlled close
  - Automatic (computer actuated) controlled close
  - Block control when parameters are out of bounds
  - Overcome communication (and other) delays by estimating end-to-end delays and predicting closing time
- And to demonstrate PMU-based control in a simple, yet pragmatic, value-adding application

Functional Requirements

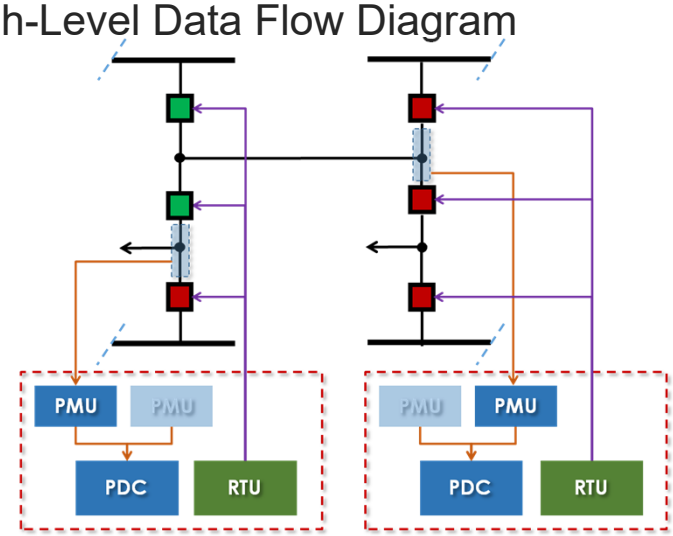
- The analytic should consist of a GUI-based application external to the openECA platform – primarily a client visualization tool – an *.exe that runs on a remote machine and is not modeled as an adapter within the openECA architecture.
- The analytic should be able to connect to a stream of synchrophasor data from the openECA platform and send control signals back to the openECA from a remote location.
- The application should have a graphical user interface (GUI) to enable ease of use by non-SME personnel. The GUI should be developed using XAML and C# using the MVVM design pattern.
- The application can utilize a 3rd party library for visualization/animation of the graphs and charts.

Functional Requirements

- The analytic must be able to estimate all major contributing end-to-end delays and provide statistical variances for each type of delay including but not limited to:
  - Measurement delivery delay (t, $\sigma_{latency}$)
  - Central concentration delay (t, $\sigma_{concentration}$)
  - Visualization delay (t, $\sigma_{visualization}$)
  - Calculation delay (t, $\sigma_{calculation}$)
  - Human actuation delay (t, σ_{human})
  - SCADA delay (t, σ_{SCADA})
  - Communication delay (t, $\sigma_{communication}$)
  - Breaker closing time delay (t, $\sigma_{breaker}$)
- The analytic must have a mechanism for user-specific configurations to account for the human actuation delay.

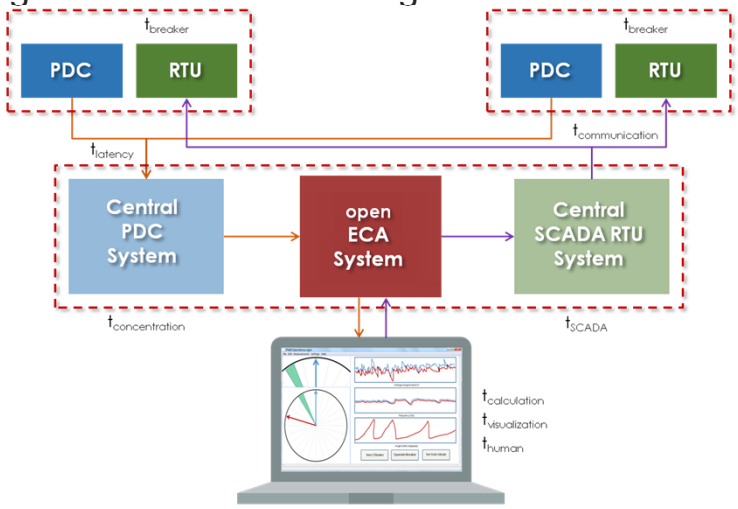
Functional Requirements

- High-Level Data Flow Diagram



Functional Requirements

- High-Level Data Flow Diagram



Functional Requirements

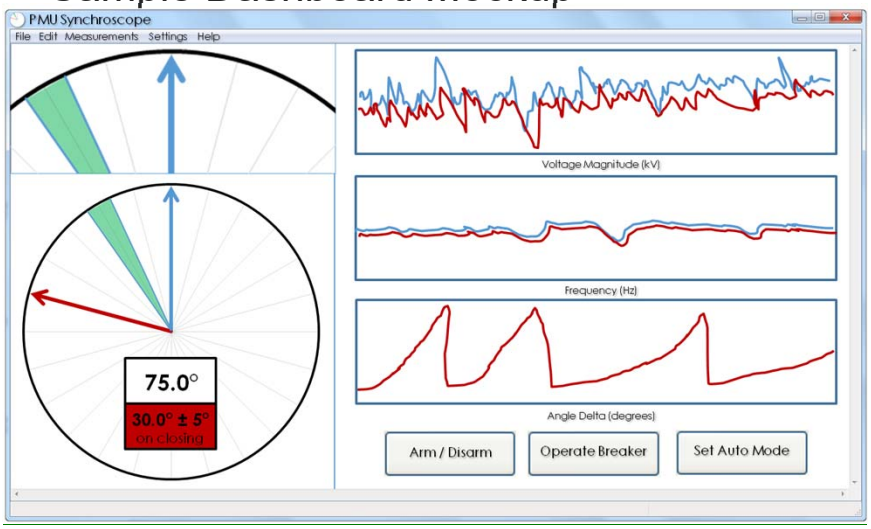
- The analytic must be able to use the combination of delays and statistical variances to form a total delay and variance that can be utilized for predicting the final closing time of the synchronization.
- The analytic must be able to predict the final closing time of the breaker during synchronization within an accuracy that is to be determined by research carried out in this project.
- The analytic must display current (real-time) estimations of the delays within the application.
- The analytic should have a dashboard where primary monitoring of signals (angles, frequency, and voltage) takes place and from where control actions could be initiated.

Functional Requirements

- The dashboard must include a polar chart that plots the relative angles across the breaker of interest.
  - The polar chart should plot a stationary phase angle (the angle from one side of the breaker referenced to itself) and a moving phase angle (the angle of one side of the breaker referenced to the other).
  - The polar chart should plot a moving window (a phase angle with a variance) that represents the most likely possible phase angle difference once the breaker actually closes.
  - The zero angle should be vertical (where the typical 90 degree mark is)
  - The polar chart should be shown in full but also provide a mechanism for viewing the fine details around the zero angle mark.
- The dashboard must include trending or strip charts of the primary monitoring signals (angle, frequency, and voltage)
- The dashboard must include buttons for executing control actions.

Functional Requirements

- Sample Dashboard Mockup



Functional Requirements

- The analytic must provide a GUI mechanism for selecting the signals to be used by the synchroscope.
- The analytic must provide a file-based mechanism for loading pre-configured signal selections so the signal selection process can be bypassed.
- The analytic must provide a file-based mechanism for saving the current signal selections so that they can be reloaded at a later time.
- The signal selection mechanism should allow for configuration that takes advantage of signal redundancy.
- The analytic must be capable of connecting to the openECA upon startup and provide GUI and file-based mechanisms for managing the connection parameters for connecting to the openECA.

openECA Provided Requirements

- The analytic requires several standard and non-standard items from the openECA platform:
  - The analytic requires the openECA to manage all data concentration tasks
  - The analytic requires the openECA to provide a connection for streaming data to a remote machine.
  - The analytic requires that the openECA provide mechanisms to streamline the signal-selection process on the client-side.
  - The analytic requires that the openECA provide any performance statistics that it has by default, or is capable of producing to the analytic for estimating time delays.

External Functional Requirements

- The analytic will require access to certain functions and networked machines to be able to estimate some of the time delays.
  - SCADA delays
- The analytic will require a 3rd party library for visualization.

Expected Computational Resources

- RAM (Low)
- CPU (Medium)
- File System (Low)

Testing and Demonstration

- Each component of the time delay estimation needs to be tested and demonstrated separately in both a lab environment and in the final demonstration environment.
- Because it is unlikely that conditions will arise in a real-time environment for this to be demonstrated on-demand, the RTDS will be the primary simulation too for testing and demonstration. Synchrophasor data will be generated from the RTDS model and streamed to an openECA instance. The analytic will connect to the openECA and execute manual and automatic controls. The resultant control signals will be sent back to the openECA and subsequently back to the RTDS.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Off-Line, Periodic
Prototype-Grade

D. Real-Time Impedance Calculator

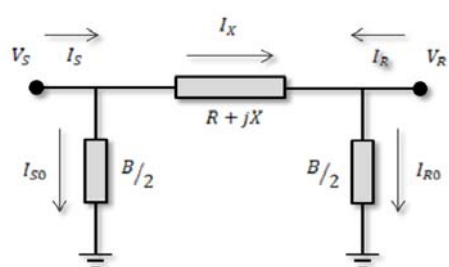
VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **Real-Time Impedance Calculator** is:
  - To compute the impedance of a series network element in an electric power system based on a two-port Π-model using voltage and current synchrophasor data from the sending and receiving ends of the series network element.
  - To compute the said impedance in real-time utilizing a stream of time-aligned synchrophasor data.
  - To serve as a supplemental troubleshooting tool for other analytics that both calculate and utilize network impedance values.
  - To serve as a simple software development example for analytic developers

Analytic Algorithm and Methodology

- The analytic should compute the impedance values for a two port pi-model of a series network element.



$$jB = 2 * \left(\frac{I_S + I_R}{V_S + V_R}\right)$$

$$\frac{V_S^2 - V_R^2}{I_S V_R - V_S I_R} = R + jX$$

Functional Requirements

- The analytic should be developed in C#
- The analytic should be developed with the latest version of .NET that is compatible with the openECA platform.
- The algorithmic components of the analytic should be developed as stand-alone libraries while the interface with the openECA platform should be developed as an Action Adapter.
- The analytic should utilize the 'Settings' dictionary mechanism to configure the adapter as much as possible. The analytic should de-serialize a configuration file for additional settings that cannot be succinctly or clearly expressed as adapter 'Settings'.

Functional Requirements

- The analytic should meet all criteria stated in its purpose.
  - The analytic should compute the impedance of one series network element
  - The analytic should utilize streaming phasor data as input and compute the impedance faster than real-time.
  - To compute the impedance values for multiple series network elements, multiple instances of the analytic adapter should be modeled, each with unique configurations.
- The analytic should accept as input, four phasor values (8 measurements)
  - V_{1mag}, θ_{V1}, I_{1mag}, θ_{I1}, V_{2mag}, θ_{V2}, I_{2mag}, θ_{I2}
- The analytic should return as output three scalar values
  - R, X, B
- The analytic should also return as output a flag that indicates the quality of validity of the three scalar values.
- The analytic should handle data quality and availability issues intelligently.

openECA Provided Requirements

- The **Real-Time Impedance Calculator** analytic requires the following functionality from the openECA platform.
  - Requires time aligned measured values per adapter configuration
  - Requires an output/calculated value delivery mechanism
  - Requires the use of adapter settings for configuration
  - Requires the ability to send commands to the adapter
  - Requires the ability to receive status updates from the adapter

External Functional Requirements

- This section is meant to describe the development/functional requirements that will not be met by the platform but are still required for the analytic to function.
- For example, the Topology Estimator requires integration with the LSE and a network model in order to function.
- In general, this section and the previous section are meant to relieve any misconceptions between the platform developers and the analytic developers regarding the interaction of the platform and the analytics.

Expected Computational Resources

- RAM (Low)
- CPU (Low per adapter, Medium if many adapter instances)
- File System (Low)

Testing and Demonstration

- Algorithmic components developed as libraries will be tested using archived PMU data from a CSV file and compared against known values.
- Adapter components will be tested by streaming archived data through the analytic and comparing the results with the aforementioned test results.
- New features of openECA that improve development and troubleshooting will be utilized when possible for the aforementioned steps.
- The final product will be tested in a real-time environment with real-time streaming synchrophasor data from the field.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time, Control
Prototype-Grade

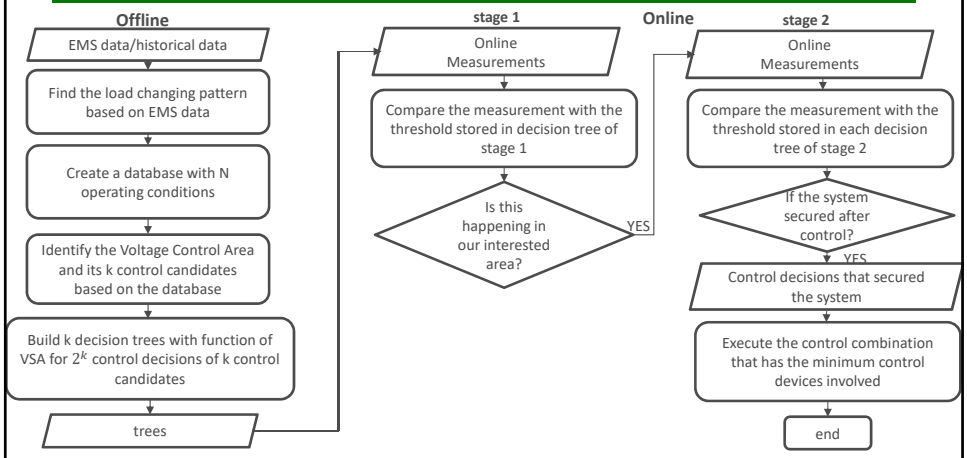
E. Regional Voltage Control

VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **Regional Voltage Control** is:
  - To provide a wide area visualization tool for situation awareness based on time-aligned synchrophasor data from PMU and Linear State Estimator.
  - To provide accurate and fast control decisions for managing voltage with LTC, series/shunt capbanks, and statcom
  - To provide cost-effective voltage control decisions that satisfying the voltage security while having minimum control devices involved (e.g. minimum number of capbanks switched).

Analytic Algorithm and Methodology



The analytic includes:

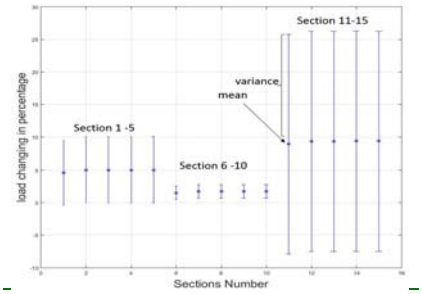
1. a offline application for database creation and decision trees training.
2. a online decision tree to identify the locations of the voltage security problem.
3. a online Parallel trees that providing voltage security assessment for control decisions

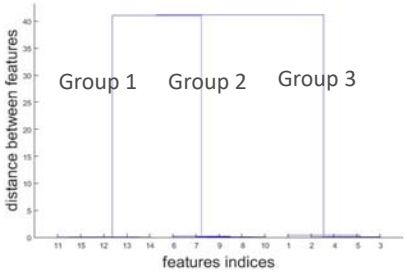
Analytic Algorithm and Methodology

Loading Changing Pattern Identification and Dimension Reduction

This offline application should be able to sample the loading conditions for each bus in the system model based on its previous year EMS data. The offline analytic should be able to identify the load changing patterns and reduce the loading conditions dimension by using feature selection technique.

Example: The 300 bus system are separated into 15 sections. Some of the sections have similar loading distribution as can be seen in the first figure. Using feature selection technique is able to cluster the sections having similar loading distribution as one group which is shown in the second figure (hierarchical clustering tree). The operating condition **P**'s dimension is reduced from 15 to 3.





Analytic Algorithm and Methodology

Loading Changing Pattern Identification and Dimension Reduction

The feature selection technique is illustrated as follows:

The feature similarity measure evaluates subsets of features to find the best feature subsets contain features highly correlated. A feature is said to be redundant if one or more of the other features are highly correlated with it. Intuitively, this correlation coefficient measures the information two features sharing. The methodology can be divided into the following steps:

Step 1: Compute the Maximal Information Compression Index between each pair of features:

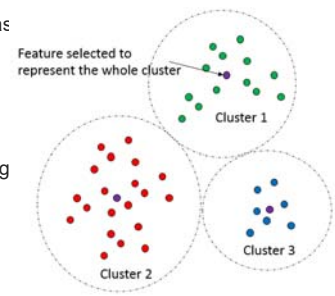
$$\mathrm{I}(x,y) = \mathrm{var}(x) + \mathrm{var}(y) - \sqrt{\left(\mathrm{var}(x) + \mathrm{var}(y)\right)^2 - 4\mathrm{var}(x)\mathrm{var}(y)(1-\rho(x,y)^2)}$$

where $\rho(x,y) = \mathrm{cov}(x,y)/\sqrt{\mathrm{var}(x)\mathrm{var}(y)}$

I is zero when the features are linearly dependent and increas as the amount of dependency decreases.

Step 2: Compute the maximum information compression index $m \times m$ matrix S which shows the relationship dependency between each features.

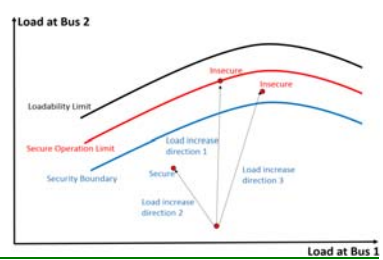
Step 3: Create K Nearest Neighbor using hierarchical clustering and find the feature closest to its nearest neighbor in each cluster as can be seen in the figure.



Analytic Algorithm and Methodology

Voltage Security Assessment

This offline application should be able to provide voltage security assessment for all operating conditions (OC) which some of them are sampled based on the EMS data and some of them are determined by operators (which might be extremely high that hitting the loadability limit). The operating conditions are considered as unstable operating conditions if the power flow diverge in PSSE, the eigenvalues of its reduced Jacbobian matrix derived by Modal Analysis is less than 0. The operating conditions considered as unstable would form a boundary which is called **secure operation limit**. Any operating conditions $\mathbf{P}$ having the shortest Euclidean distance $\|\mathbf{P}_* - \mathbf{P}_0\|$ to the **secure operation limit** ($\mathbf{P}_*$ is the operating condition lying at the secure operation limit) is less than a threshold value , N-1 contingency unstable, or any bus voltage is lower than a threshold value (0.95 p.u) would be considered as insecure operating conditions. These operating conditions would form a new boundary: **insecure boundary**. For example, if the system has two load buses only, the boundary can be viewed as following figure. As it can be seen, the system will operate towards different directions, however, once its operating condition reaches the **security boundary**, this operating condition is insecure.

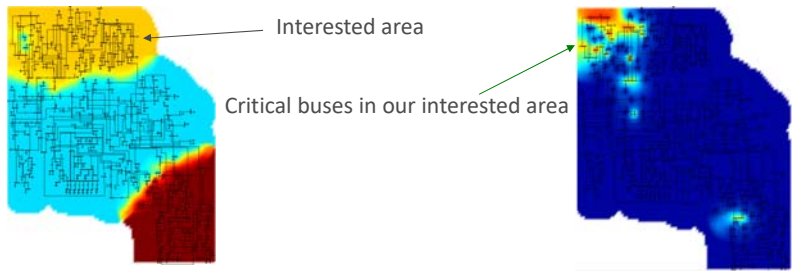


Analytic Algorithm and Methodology

Voltage Control Area Identification

The offline application can find the voltage control area (VCA) and control candidates for our interested area. The interested area is the region of power system where the utilities concern the most. It is known that it will be complicated to build DTs for numerous control decisions when the number of reactive power support control variables are tremendous. If just simply consider capbank, for a power system with 80 capbank for reactive power control, the control combinations of capbank switching will be 2^{80}. It will be a huge burden for learning sample generation and DT training; besides, due to the high reactance in transmission line, some of the control would not impact our interested area.

Voltage control area (VCA) is determined by the critical buses inside our interested area. These buses can be found out by using **Modal Analysis** based on the insecure operating conditions. For instance, in IEEE 300 buses system, assuming our interested area is area 2 (yellow), the critical buses (bright color) can be seen in the right figure. The methodology is introduced in the next 3 slides.



Analytic Algorithm and Methodology

Voltage Control Area Identification

Modal Analysis is introduced as follows:

For a given OC, the network constraints are expressed in the following linearized model:

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = \boldsymbol{J}\begin{bmatrix}\Delta\theta\\ \Delta V\end{bmatrix}$$

$$\boldsymbol{J} = \begin{bmatrix}J_{P\theta} & J_{PV}\\ J_{Q\theta} & J_{QV}\end{bmatrix}$$

where: ΔP – incremental change in bus real power; ΔQ – incremental change in bus reactive power

$\Delta\theta$ – incremental change in bus voltage angle; ΔV – incremental change in bus voltage magnitude; $\boldsymbol{J}$ – Jacobian matrix

By letting $\Delta P = 0$, we can have:

$$\Delta Q = J_r * \Delta V$$

where J_r is the reduced Q-V Jacobian sub-matrix

$$J_r = [J_{QV} - J_{Q\theta} J_{P\theta}^{-1} J_{PV}]$$

Since J_r can be written as:

$$\Delta V = J_r^{-1} * \Delta Q$$

the inverse matrix J_r^{-1} can be computed based on V-Q sensitivity matrix:

$$J_r^{-1} = |\partial V / \partial Q|$$

Analytic Algorithm and Methodology

Voltage Control Area Identification

The ith diagonal element of matrix J_r^{-1} is the V-Q sensitivity at bus I, which represents the slop o the Q-V curve at the given operating point. A positive V-Q sensitivity is indicative of stable operation the smaller the sensitivity the more stable the system. The sensitivity becomes infinite at the stability limit.
The relationship between bus V-Q sensitivities and eigenvalues can be derived from the general equation:

$$J_r = \boldsymbol{\xi \Lambda \eta}$$

$$\boldsymbol{\xi} = [\xi_1, \xi_2, \dots, \xi_n]$$

$$\boldsymbol{\eta} = [\eta_1, \eta_2, \dots, \eta_n]^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix}$$

where $\boldsymbol{\xi}$and $\boldsymbol{\eta}$ represent the left and right eigenvectors of V-Q sensitivity matrix while λ_i denotes the eigenvalue of J_r . The V-Q sensitivity at bus k given by:

$$\frac{\partial V_k}{\partial Q_k} = \sum_i \xi_{ki}\, \eta_{ik} / \lambda_i$$

The V-Q sensitivities provide information regarding the combined effects of all modes on V-Q variation. The magnitude of the eigenvalues can provide a relative measurement of the proximity to voltage instability. When the system reaching the security boundary, the modal analysis is able to identify the critical bus which participating in each critical mode (there will be more than one critical mode). The relative participation of bus k in mode i is given by the bus participation factor (PF):

$$PF_{ki} = \xi_{ki}\eta_{ik}$$

Analytic Algorithm and Methodology

Voltage Control Area Identification

Pick the buses with PF higher than a predefined threshold value PF* to form a set for further analysis (SFA). One operating condition (OC) corresponds to one SFA_i

Cluster the OC based on the similarity of all SFA. All the buses associated the clustered OC are forming a VCA. It is noted that each identified VCA is associated with the clustered OC; these OCs are so-called "**support operating condition**" of that VCA. Therefore, we can classify the OCs for different identified VCA:

Voltage Control Area	Support Operating Condition
VCA_1	$OC_1, OC_2, OC_4, \dots\dots$
VCA_2	$OC_3, OC_5, OC_6, \dots\dots$
…	…
VCA_m	$OC_7, OC_8, OC_{10}, \dots\dots$

Analytic Algorithm and Methodology

Control Candidates Selection

Applying each single control to all of those "**support operating conditions"** associated with our interested area and see their participation factor change.

- For example, one capbank is switched on for all insecure operating conditions
- Compute the participation factor reduction at all critical buses for insecure operating condition k

$$dPF_k = \sum_{i=1}^{m} \frac{PF_{i,k}^{after} - PF_{i,k}^{before}}{\max(PF_{1,k}^{before}, PF_{2,k}^{before}, \dots, PF_{m,k}^{before})}$$

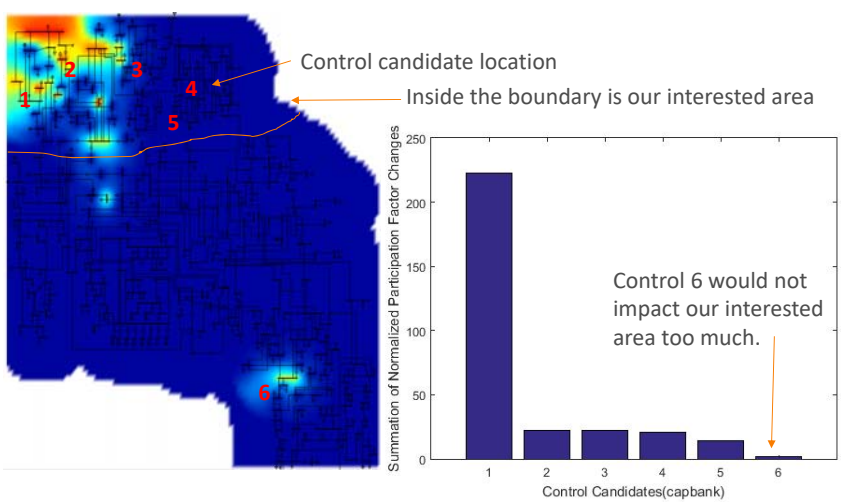
$PF_{i,k}^{before}$: PF before any control ; $PF_{i,k}^{after}$: PF after one single control

$$PF\ reduction\ for\ control\ n^{th} = \sum_{k=1}^{K} dPF_k$$

The voltage control device has minim PF reduction indicates that this control would not impact our interested area. For example: in IEEE 300 bus system, there are 6 fixed capbanks. Based on the PF reduction for each of these capbank, we can eliminate control candidate 6 (see next page).

Analytic Algorithm and Methodology

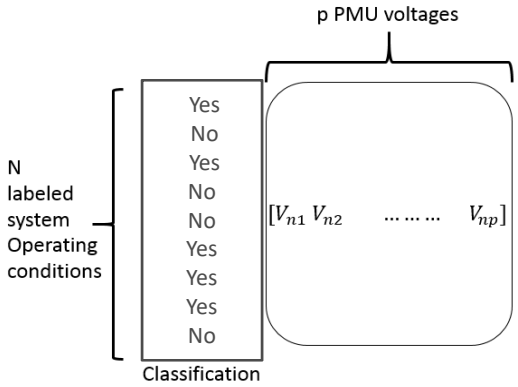
Control Candidates Selection



Analytic Algorithm and Methodology

Stage 1 Decision Tree for Security Problem Identification

This decision tree is trained to determined if the voltage security problem is happening in our interested area. If yes, the next stage for voltage control will be activated. The database format can be seen as follows: If the operating condition is the "**support operating conditions"** associated with the VCAs inside our interested area, the operating conditions will be labeled as yes, the rest of the operating conditions are labeled as NC

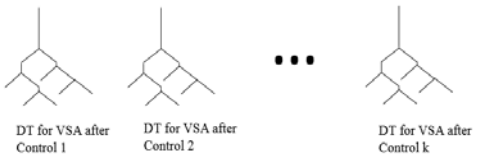


Analytic Algorithm and Methodology

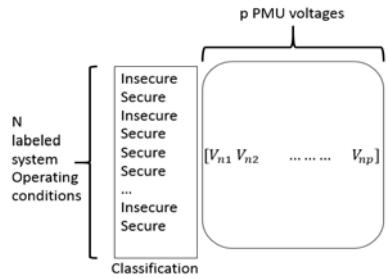
Stage 2 Decision Tree for Security Problem Identification

All the **selected control candidates**, **insecure operating conditions/support operating conditions** associated with our interested area, and all **secure operating conditions** are saved in database for decision tree (DT) offline training.

- Each control decision is a combination of the selected control candidates. If there are 5 control candidates, there will be 2^5 control decisions. If control candidate 1 is switched on but the rest are off, the control decision will be [1,0,0,0,0]
- Each control decision will train a decision tree. Each decision tree is able to provide voltage security assessment.
- There will be 2^5 parallel decision trees.



Parallel Decision Trees



Database format for each tree

Analytic Algorithm and Methodology

Stage 2 Decision Tree for Security Problem Identification

Providing there is an insecure OC:

$$voltage\ measurements\ pu = [1.024, 1.0312, 0.989 \ldots \ldots \ldots, 1.0138]$$

- Assuming this OC has activated the second stage.
- 32 trained decision trees will operated simultaneously and determine which decisions can secure the system. The simulation result is shown as follows:
- Insecure Decisions: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 32]
- Secure Decisions: [16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31]
- As it can be seen, the secure decisions with minimum devices involved are:
  Decision 16 : [0,1,0,0,0,0] It is suggested to switch capbank 2
- Secure decision such as 17: [0,1,1,0,0,0] has more than one device involved.

Analytic Algorithm and Methodology

Notes of the Methodology

- This analytic is highly system dependent.
- The effectiveness of the analytic is highly depending on the created databases.
- It is noted that the database is also determined by scheduled topology change (not contingencies).
- It is suggested that the database for decision tree training can be updated regularly or it can be updated when there is a scheduled system topology change.

Functional Requirements - online

- The analytic should consist of two components: (1) an on-line adapter and a (2) off-line application.
- The on-line adapter includes two stages decision tree for voltage control.
- The on-line adapter should assess each frame of the streaming synchrophasor data.
- The adapter should be developed in C# using the latest version of .NET support by openECA and the Grid Solutions Framework.
- The adapter should utilize the 'Settings' dictionary mechanism to configure the adapter as much as possible. The analytic should de-serialize a configuration file for additional settings that cannot be succinctly or clearly expressed as adapter 'Settings'.
- The adapter should accept as input, (1) positive sequence phasors (V) from all PMUs and any topology related signals such as breaker statuses.
- The adapter should generate integer-based output signals that should be published to the openECA platform and archived along side all PMU data.

Functional Requirements - offline

- The offline application should encapsulate all functionality of the offline methodologies: Loading Changing Pattern Identification, Voltage Control Area Identification, Control Candidates Selection.
- The offline application should be developed in python and MATLAB.

openECA Provided Requirements

The **Regional Voltage Control** analytic requires the following functionality from the openECA platform.

The on-line adapter requires:

- Requires time aligned measured values per adapter configuration
- Requires an output/calculated value delivery mechanism
- Requires the use of adapter settings for configuration
- Requires the ability to send commands to the adapter
- Requires the ability to receive status updates from the adapter

External Functional Requirements

- The off-line application may utilize MATLAB or Python scripts to perform some types of data manipulation.
- The off-line application and the on-line adapter will utilize the same network model (and measurement mapping) that is used by the Linear State Estimator.

Expected Computational Resources

- RAM (Low for real-time, High for off-line)
- CPU (Low for real-time, High for off-line)
- File System (High for large systems during decision tree training)

Testing and Demonstration

The testing and demonstration are divided into 3 phases.

Phase 1: Demonstration using IEEE model and simulated loading conditions.

This testing will verify all methodologies implemented in the analytic. Currently for this phase, the methodologies (except stage 1 online decision tree) implemented in this analytic had already being verified with IEEE 118 and IEEE 300 bus system.

Phase 2: Demonstration using Dominion system model, archived previous year 3 months summer load data in Dominion network.

In this phase, the offline application will provide information of loading pattern, voltage control area, and control candidates for Dominion system. The voltage measurements will be provided by PSSE simulation based on 3 months summer load data in Dominion network. Success at this phase is characterized by providing accurate control decisions for all of these real system loading conditions. This phase doesn't require demonstration with openECA.

Phase 3: Demonstration using Dominion system model, archived 3 month summer load data in Dominion region, and simulation in RTDS

In this phase, the RTDS will model the identified voltage control area given by phase 2 (which should be much smaller than the whole Dominion network). Voltage measurements, breaker status information, and capbank/LTC/statcom status provided by RTDS will be streamed in to openECA. Success at this phase is characterized by providing accurate control decisions in RTDS for all of the real system loading conditions.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time
Prototype-Grade

F. Topology Estimator

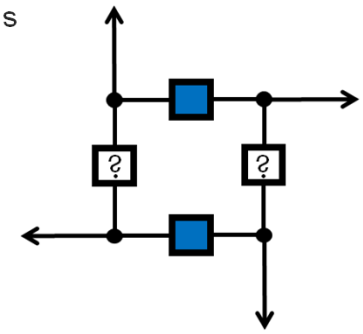
VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **Topology Estimator** is to provide increase awareness of the real-time network topology to the *Linear State Estimator* and other network applications.
- Using measurements of voltage phasors, current phasors, and circuit breaker statuses from across the transmission network, the topology estimator should improve knowledge of current topology by:
  - Observing unmeasured breaker statuses
  - Validating measured breaker statuses
  - Identifying topological errors
  - Inferring state of unmetered equipment
  - Reducing impact of topological discrepancies on the LSE

Analytic Algorithm and Methodology

- Problem Statement - Breaker status and topological information is key for accurate topology processing used in the LSE and other applications.
  - Breaker status in PMU streams not ubiquitous
  - Time tagged breaker status not standardized.
  - Breaker status sometimes supplemented by SCADA.
  - Breaker statuses susceptible to latching or other measurement error.



Analytic Algorithm and Methodology

- Research will be performed to attempt to determine a closed-form solution to the problem.
- However, the **Topology Estimator** methodology will likely be comprised of a set of complementary functions that, when used together, enable the desired overall functionality.

Analytic Algorithm and Methodology

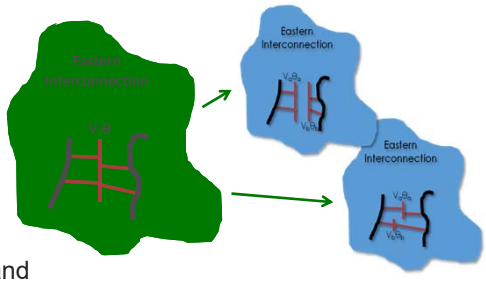
- This collection of functions will likely include:
  - Determination of voltage phasor coherency (equipotential) for multiple substation voltage phasors in the same voltage class
  - The application of Kirchhoff's Current Law as a test for topological completeness
  - The inference of the status of series and shunt compensation devices
  - A binary determination of current flow on a series element using pre-defined thresholds. A '1' is for flow above the given threshold and implies that the series network element is energized.
  - Detection of a topological state change using previously developed predictive algorithms.

Analytic Algorithm and Methodology

- Determining Voltage Coherency
  - Two phasors that have been deemed valid and are of sufficient quality can be considered *'coherent'* if they are empirically determined to be at the same equipotential.
  - How can this be known?
    - One option is to check if TVE between V_1 and V_2 is less than a pre-determined threshold assigned via offline studies (see '*How to Find* $\delta_{coherency,i}$,' in later slides).
    - Another option is to assume a completely connected bus and perform an isolated estimation problem on the bus voltages. Then a Nearest Neighbor Search of the residuals should yield a grouping then establishes which voltage phasors are coherent.
    - Note that since there will be less variance in voltage angle among coherent voltages, the hypothesis is that ΔV_θ will be more telling than ΔV_{mag} for detecting coherency
    - Also note that de-energized nodes cannot be coherent with any other node, not even other de-energized nodes.

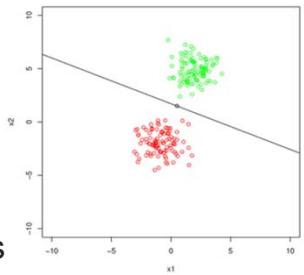
Analytic Algorithm and Methodology

- *How to Find* $\delta_{coherency,\,i}$?
  - An expected threshold for each, i, bus can be determined through a series of off-line power flow analyses.
  - This can be executed both on the MMWG model of the Eastern Interconnection as well as on operational, nodal, models of the DVP (and surrounding) system.
  - To achieve this:
    - Procedurally split each bus into each of its possible configurations one at a time (one change per power flow)
    - For each power flow, observe the change in complex voltage at the bus of interest for pre- and post-load flow conditions.
    - Vary daily and seasonal load and repeat



Analytic Algorithm and Methodology

- *How to use Nearest Neighbor Search of residuals*?
  - Run an isolated state estimation problem for the substation of interest using only voltage phasors and assuming that all circuit breakers and switches are closed.
  - Compute the WLS solution and associated residuals for each phasor measurement.
  - Then use Nearest Neighbor Search to group the phasor measurements into possible coherency groups.



Analytic Algorithm and Methodology

- Topology Estimation via Voltage Coherency
  - There are some basic rules which drive the logic used to assess topology based on voltage coherency.
    - The presence of a valid, good quality voltage phasor (and the case of a line flow, a current phasor also works) indicates that a node is *energized*. This transforms the analog measurements into a binary value based on a threshold
    - The energized nodes with voltage phasors that are considered to be coherent form *coherency groups*.
    - A path that is traced from across all nodes in a given coherency group is called a *coherency path*. A single coherency group may have multiple coherency paths. The set of coherency paths represent all of the possible solutions for a given bus.

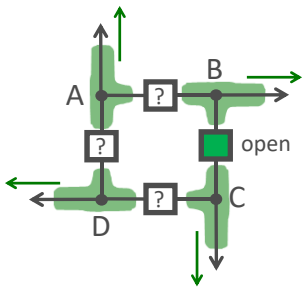
Analytic Algorithm and Methodology

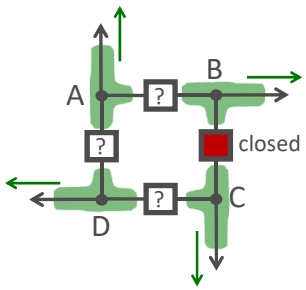
- Topology Estimation via Voltage Coherency
  - There are some basic rules which drive the logic used to assess topology based on voltage coherency.
    - Breaker statuses are considered to be correct unless indicated otherwise by the PMU Status Word (or other similar mechanism). Therefore, knowledge of a subset of breaker status in a given bus configuration can limit the solution set. *See next slide*.
    - Valid, good quality phasor measurements take precedence over a breaker status when determining topology and therefore, enable determination of breaker status measurement error. *See the slide after next*.
    - Coherency groups are equivalent to *ObservedBusses* in the Linear State Estimator and can be substituted as such. Therefore, the LSE minimizes its dependency on breaker status measurements to perform topology processing due to the information provided by the **Topology Estimator**. *See the slide after next, also*.

Analytic Algorithm and Methodology

Knowledge of subset of breaker status can limit solution set.

Coherency Group:
{A, B,C, D}





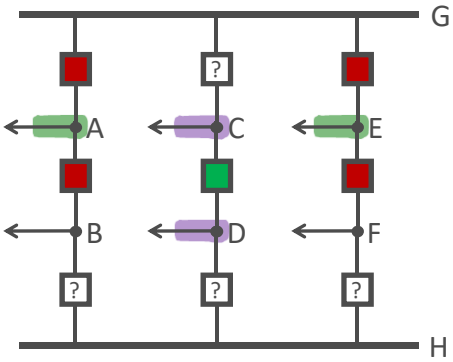
Coherency Paths:

{A, B, C, D} → {[A↔B↔C↔D], [B↔C↔D↔A], [C↔D↔A↔B], [D↔A↔B↔C]}

{[A↔B↔C↔D], [B↔C↔D↔A], [C↔D↔A↔B], [D↔A↔B↔C]}

Analytic Algorithm and Methodology

Detecting Errors in Topology Measurements - The **Topology Estimator** can reduce the network to buses like the Topology Processor can. When these two don't match, there is an error in assumed topology by the Topology Processor.



Coherency Groups: {A, E }
{C, D}

Topological Groups: {A, B, G, E, F }
{C}
{D}

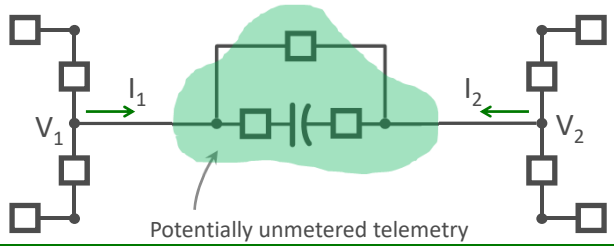
Therefore, $CB_{C\text{-}D}$ breaker status measurement is incorrect. *Additionally*, TE Coherency Groups can be substituted for Observed Buses in LSE bus-branch form.

Analytic Algorithm and Methodology

- Series Compensator Status Inference
  - In order for the Linear State Estimator to include a series compensated transmission line in the state equation, it must know the total Z of the line, included the series compensation device.
  - Availability of telemetry is a issue for series compensated devices. Both breaker statuses and/or compensator telemetry may be unavailable.

Analytic Algorithm and Methodology

- Series Compensator Status Inference
  - With sufficient measurements on either end of the series compensated transmission line, the total Z of the line can be computed in real-time.
  - Using the computed Z value, the status of the device an be inferred.



Analytic Algorithm and Methodology

- Detection of a Topology Change
  - The ability to detect a change in topology (without knowledge of what change occurred) is often useful for several reasons:
    - For the validation of measurements
    - For the reduction of computation burden (no need to run the complete Topology Estimator or Topology Processor if no topology change is detected).
  - How to detect a change in topology?
    - When a phasor measurement moves from below a pre-defined threshold to above (or vice versa.
    - When there is a change in the coherency groups or paths.
    - Use of measurement prediction.

Functional Requirements

- The analytic should function for both positive sequence and three phase configurations of the Linear State Estimator.
- The analytic should take advantage of any measurement redundancy from three phase monitoring to perform its functions.
- The analytic should consist of four (4) components:
  - The computational component, developed in C# using the latest version of .NET that is compatible with openECA and the GSF, directly integrated with the libraries of the LSE. This component primarily encompasses the aforementioned methodologies.
  - An on-line adapter that will either be directly integrated with the LSE adapter or function alongside the LSE adapter (C#, .NET, GSF). This will run directly inside of the openECA.
  - An off-line GUI-based module (including the LSE) used for testing and debugging using archived data and streaming data from openECA.
  - An off-line GUI-based tool, integrated with the LSE network model builder, that enables any necessary modeling and configuration for the Topology Estimator.

Functional Requirements

- The computation component of the analytic should provide all of the functionality described in the algorithms and methodology section
- The computation component of the analytic should provide any additional functionality not mentioned in the algorithms and methodology section yet later deemed necessary to achieve the desired behavior.
- All component functionality should be configurable/toggle-able using file-based settings.

Functional Requirements

- The on-line adapter should (1) provide the interface between the openECA and the algorithms contained in the libraries and (2) utilize the high level functions provided by the libraries to perform the Topology Estimation functions.
  - The adapter should be developed in C# using the latest version of .NET support by openECA and the Grid Solutions Framework.
  - The adapter should utilize the 'Settings' dictionary mechanism to configure the adapter as much as possible. The analytic should de-serialize a configuration file for additional settings that cannot be succinctly or clearly expressed as adapter 'Settings'.
  - The adapter should accept as input, (1) positive sequence phasors (V & I) and/or A, B, & C phase phasors from all PMUs and (2) any Status Word and data quality information available for the aforementioned phasors and (3) any topology related signals such as breaker statuses.
  - The adapter should generate topological information that can be (1) directly (in code) handed off to the Linear State Estimator and (2) converted into a Time Series data point and output for archive or publishing.
  - The adapter must have the ability to instantiate a set of initial 'known' breaker statuses using an XML file based format.

Functional Requirements

- The offline-module (with LSE functionality) should be able to connect to the openECA and stream data directly to it for testing and troubleshooting purposes.
- The offline-module should be able to digest file-based snapshots of synchrophasor data for testing and troubleshooting purposes.
- The offline-module should have a graphical user interface (GUI) to enable ease of use. The GUI should be developed using XAML and C# using the MVVM design pattern.

openECA Provided Requirements

- The **Topology Estimator** analytic requires standard functionality from the openECA platform.
- The on-line adapter requires:
  - Requires time aligned measured values per adapter configuration
  - Requires an output/calculated value delivery mechanism
  - Requires the use of adapter settings for configuration
  - Requires the ability to send commands to the adapter
  - Requires the ability to receive status updates from the adapter
- The off-line applications require:
  - The ability to directly connect to the openECA and stream synchrophasor data to the off-line applications in an remote environment for testing and troubleshooting.

External Functional Requirements

- The analytic will require direct integration with the Linear State Estimator and the LSE code base.
- The analytic will require modifications to the network model schema used by both the analytic and the LSE.
- The analytic will require an updated network model with associated measurement mapping (just like the LSE) in order to function.

Expected Computational Resources

- RAM (Low)
- CPU (High)
- File System (Low)

Testing and Demonstration

- Initial testing and development of the analytic and its components will be performed using data synthesized from load flow solutions. Functionality will be tested in the absence of measurement error to ensure proper behavior. Success during this stage will be characterized by both a topology and a state solution which are equivalent to the simulated data.
- Additionally testing will be performed on synthesized data sets from load flow solutions where measurement error and data loss has been introduced. Success during this stage will be characterized by a topology and a state solution which are statistically equivalent to the unmodified simulated data and behave robustly in the presence of data loss.

Testing and Demonstration

- Following this, the RTDS and Dominion network model will be used to generate real-time streaming phasor data to an openECA test environment where the analytic can be exercised. Success during this stage will be similar to the error-less simulated data test. However, this will demonstrate the ability of the analytic to function at scale in real-time.
- Next, the analytic will be tested using off-line archived PMU data from the Dominion system with a known solution. Success during this stage will be similar to the error-induced simulated data test. However, the use of real synchrophasor data will further exercise the nuances of the analytic and its network model.
- Finally, the analytic will be tested and demonstrated in a real-time environment in the openECA with streaming synchrophasor data from the Dominion system. Success during this stage should encompass the positive results from all previous tests

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Off-Line, Periodic
Prototype-Grade

G. Transmission Line Impedance Calibration

VirginiaTech Dominion

Purpose of the Analytic

- The purpose of the **Transmission Line Impedance Calibration** analytic is:
  - To use synchrophasor measurements to improve the accuracy of transmission line impedances and therefore, (1) reduce the impacts of these types of topology errors on network applications and (2) provide validation of calculated impedances used in system analysis.
  - This is accomplished by estimating the values of transmission line impedances under certain constraints identified in the methodology.

Analytic Algorithm and Methodology

- The analytic should compute estimated transmission line impedance values for transmission lines sufficiently metered by PMUs in the Dominion network. This is subject to:
  - The presence of at least one high quality voltage phasor measurement from a revenue quality PT and one high quality current phasor from a high accuracy CT (i.e. the PT and CT design should minimize RCF and PACF so that the resultant phasor can be assumed to be a perfect measurement)
  - The ability to form a tree of measurements connecting all transmission lines that have sufficient PMU measurements for the impedances to be estimated.
  - The availability of periodic samples of synchrophasor data during a 24 hour period for all metering points in the tree of measurements such that many different loading conditions are included in the sample set.

Analytic Algorithm and Methodology

- The algorithms and methodology of the **PMU Instrument Transformer Calibration** analytic will be extended to include transmission line impedances by adding a high accuracy CT at the location of the high accuracy voltage measurement. With $W = 1 + yZ$, the estimated impedance matrix is:

$$\hat{Z} = \frac{1}{y(W+1)} \begin{bmatrix} W\dfrac{Ki_1}{Kv_1} & \dfrac{Ki_2}{Kv_1} \\ \dfrac{Ki_1}{Kv_2} & W\dfrac{Ki_2}{Kv_2} \end{bmatrix}$$

where y is the admittance of the shunt connections of the pi-model and Z is the impedance of the series branch.

Analytic Algorithm and Methodology

- Then, the impedance and ratio errors can be estimated simultaneously.

$$\widehat{W} = \sqrt{\frac{\hat{Z}(1,1)\hat{Z}(2,2)}{\hat{Z}(1,2)\hat{Z}(2,1)}}$$

$$\widehat{K}I(2) = \widehat{W}\frac{\hat{Z}(2,1)}{\hat{Z}(1,1)}KI(1)$$

$$\widehat{K}V(2) = \frac{1}{\widehat{W}}\frac{\hat{Z}(1,1)}{\hat{Z}(1,2)}KV(1)$$

$$\hat{y} = \sqrt{\frac{1}{det(\hat{Z})}\frac{\widehat{K}I(1)}{\widehat{K}V(1)}\frac{\widehat{K}I(2)}{\widehat{K}V(2)}\frac{W-1}{W+1}}$$

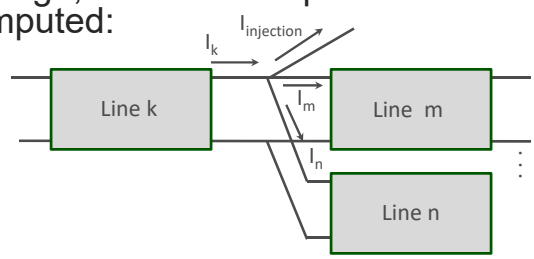
$$\hat{Z} = \frac{1}{\hat{y}}(\widehat{W} - 1)$$

Analytic Algorithm and Methodology

- However, the quantity W can be estimated with only voltage measurements. Since $y = jb$ is taken to be independent of temperature, the real and imaginary parts of $W \rightarrow W = (1 - bX) + jbR$ show the effects of temperature on the series resistance and reactance separately
- Temperatures by zip code can be obtained from the web and associated with the monitored network equipment.
- The temperature values can be included in offline calculations to learn relationships between line impedances, temperature, and line loading.

Analytic Algorithm and Methodology

- The addition of the accurate current measurement requires that all injections at a substation in the measurement tree be monitored. This may limit the application of the analytic in some systems. Unlike the voltage, the current input to the next line must be computed:



- Given I_k, I_m, and I_n, the injection, $I_{injection}$, is required.

Functional Requirements

- The functional requirements for the **Transmission Line Impedance Calibration** analytic are identical to the requirements for the **PMU Instrument Transformer Calibration** analytic (aside from the requirement to execute the aforementioned methodology).

openECA Provided Requirements

- The **Transmission Line Impedance Calibration** analytic requires the same functionality from the openECA platform that the **PMU Instrument Transformer Calibration** analytic requires.

External Functional Requirements

- External function requirements for the **Transmission Line Impedance Calibration** analytic are identical to the **PMU Instrument Transformer Calibration** analytic.

Expected Computational Resources

- RAM (Low)
- CPU (Medium in real-time, Medium off-line)
- File System (High for large systems and archived data)

Testing and Demonstration

- Because the **Transmission Line Impedance Calibration** analytic is a derivative of the **PMU Instrument Transformer Calibration** analytic, the same testing and demonstration procedures used for the **PMU Instrument Transformer Calibration** analytic should be used.
- Success will be characterized at each stage by the ability of the analytic to compute consistent an accurate values of transmission line impedances under the constraints of each test.

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time
Production-Grade

H. Oscillation Detection

T&D Consulting Engineers

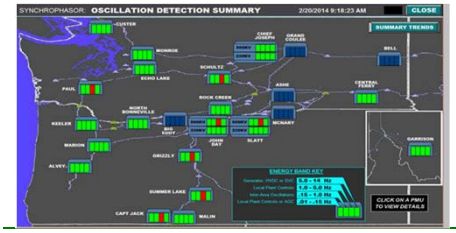
MontanaTech THE UNIVERSITY OF MONTANA

Purpose of the Analytic

- The purpose of the **Oscillation Detector** is to continuously monitor a multitude of synchrophasor inputs covering a wide geographic area for oscillations.
  - The frequency of an observed oscillation may provide some insight as to the cause of the oscillation.
  - The energy of the oscillation may provide some insight into the severity of the oscillation.
  - The shape of, or in this case the geographic coverage of, an oscillation may also indicate severity.
- Most high-energy oscillations are due to malfunctioning system components. An oscillation detector can help utilities and system operators identify and correct malfunctioning equipment.

Analytic Algorithm and Methodology

- The analytic uses simple filters to detect oscillations. Response time is important. Light CPU burden is also important because it is anticipated that this analytic will run on hundreds of input signals.
- The analytic requires user-defined configuration information.



Example of a user interface geographically depicting oscillation detector outputs.

Analytic Algorithm and Methodology

- A typical configuration for the analytic may be as follows:
  - Retrieve 200 streaming synchrophasors from openECA;
  - Combine the 200 synchrophasors into 200 “derived signals” and downsample to 5 and 30 samples per second;
  - Pass each of the 200 derived signals through four bandpass filters;
  - Return a result.

Functional Requirements

- The analytic should utilize standard openECA methods for configuration.
- The analytic should be developed as a .NET library capable of compatibility with openECA interface standards.
- The analytic should be capable of producing oscillation energy, i.e. a “result set”, at user-defined reporting rates.

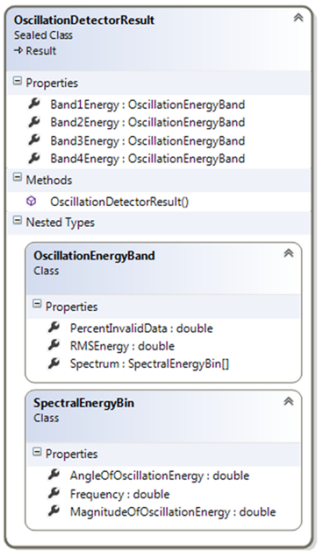
Functional Requirements

- The analytic should include in its “result set” the following fields:
  - rmsEnergy for each of four energy “bands” as double scalars
  - (magnitude, angle, identifier)[] as enumeration of FFT results
  - flags and indicators of the “goodness” of the estimate
- The analytic should accept as inputs:
  - Unfiltered MW, MVAR, VoltageAngle, Magnitude as derived from an underlying streaming data source.*

*NOTE: The input(s) to the analytic can be any linear combination of the aforementioned data types. For example, an input can be the sum of two MW signals or the difference between two voltage angles. The analytic *must* be provided raw, unfiltered inputs. The analytic should be capable of directly processing raw synchrophasors to form the desired inputs in the event openECA cannot provide appropriate “derived” inputs.

Functional Requirements

- The analytic's result set may be represented by the following diagram:



openECA Provided Requirements

- The analytic requires the following from openECA:
  - Time-aligned measured values, either phasors (two quantities per measurement) or real-valued measurements (one quantity per measurement)
  - Ability to accept abstract result sets
  - Ability to manage configuration information and to provide configuration information to the analytic from time to time

NOTE: The analytic operates on real-valued input data. In most cases the real-valued input data is derived from synchrophasor inputs. This slide is written with the assumption that openECA will provide raw synchrophasor inputs. There exists a possibility that openECA might provide "derived" inputs.

openECA Optional Features

- All other factors being equal, it may be nice if the openECA provided the following:
  - "Derived signals", i.e. linear combinations of streaming synchrophasor inputs. Some examples are:
    - Sum of two MW signals, an example of which might be V_1*conjugate(I_{42}) + V_{11}*conjugate(I_{65})
    - Difference of three weighted voltage angles, an example of which might be 4*angle(V_{11}) + 3*angle(V_{13}) – 7*angle(V_{12})

External Functional Requirements

- No known external functional requirements.

Expected Computational Resources

- RAM (Low)
- CPU
  - Light burden for each configured oscillation detector
  - Moderate burden for hundreds of oscillation detectors
  - Use of parallel threads desired
  - Possible use of unmanaged library to accelerate speed
- File System (Low)

Testing and Demonstration

- Algorithms will be tested for accuracy using data sets provided by developers in csv format
- Algorithms will be tested for robustness, e.g. ability to handle missing inputs, using data sets provided by developers in csv format

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Real-Time
Production-Grade

I. Oscillation Mode Meter

T&D Consulting Engineers

MontanaTech THE UNIVERSITY OF MONTANA

Purpose of the Analytic

- The purpose of the **Mode Meter** is to continuously estimate the frequency, damping and shape of a single *specific* system oscillatory mode.
  - The frequency of an oscillatory mode describes the rate at which energy is transferred between coherent groups of generators in a power system.
  - The damping describes the propensity of the mode to perpetuate oscillations. Low damping of a mode can be an indicator of system stress.
  - The shape of a mode defines the coherent groups of generators participating.
- Some oscillatory modes are known to be of concern with respect to bulk grid reliability. For these modes a **Mode Meter** can be used to continuously monitor the properties of the mode.

Analytic Algorithm and Methodology

- A typical configuration for the analytic may be as follows:
  - Retrieve 30 streaming synchrophasors from openECA;
  - Combine the 30 synchrophasors into 8 “derived signals” and downsample to 5 samples per second;
  - Store one hour of each of the 8 “derived signals” at 5 samples per second;
  - When required to process results, retrieve the one hour of buffered data, process it, and return a result.

Functional Requirements

- The analytic should utilize standard openECA methods for configuration.
- The analytic should be developed as a .NET library capable of compatibility with openECA interface standards.
- The analytic should be capable of producing mode estimates, i.e. a “result set”, at user-defined reporting rates.

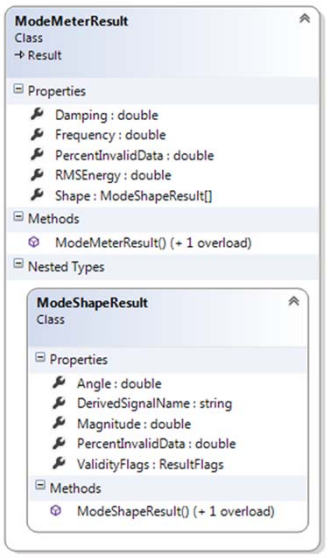
Functional Requirements

- The analytic should include in its “result set” the following fields:
  - freq, damping, rmsEnergy as double scalars
  - (magnitude, angle, identifier)[] as enumeration of shape results
  - flags and indicators of the “goodness” of the estimate
- The analytic should accept as inputs:
  - Unfiltered MW, MVAR, VoltageAngle, Magnitude as derived from an underlying streaming data source.*

*NOTE: The input(s) to the analytic can be any linear combination of the aforementioned data types. For example, an input can be the sum of two MW signals or the difference between two voltage angles. The analytic *must* be provided raw, unfiltered inputs. The analytic should be capable of directly processing raw synchrophasors to form the desired inputs in the event openECA cannot provide appropriate “derived” inputs.

Functional Requirements

- The analytic’s result set may be represented by the following diagram:



openECA Provided Requirements

- The analytic requires the following from openECA:
  - Time-aligned measured values, either phasors (two quantities per measurement) or real-valued measurements (one quantity per measurement)
  - Ability to accept abstract result sets
  - Ability to manage configuration information and to provide configuration information to the analytic from time to time

NOTE: The analytic operates on real-valued input data. In most cases the real-valued input data is derived from synchrophasor inputs. This slide is written with the assumption that openECA will provide raw synchrophasor inputs. There exists a possibility that openECA might provide "derived" inputs.

openECA Optional Features

- All other factors being equal, it may be nice if the openECA provided the following:
  - "Derived signals", i.e. linear combinations of streaming synchrophasor inputs. Some examples are:
    - Sum of two MW signals, an example of which might be V_1*conjugate(I_{42}) + V_{11}*conjugate(I_{65})
    - Difference of three weighted voltage angles, an example of which might be 4*angle(V_{11}) + 3*angle(V_{13}) – 7*angle(V_{12})
  - Data buffer, i.e. a storage location for a time series of data. An example is:
    - One hour of historical data from a "derived signal" sampled at 5 samples per second.

External Functional Requirements

- No known external functional requirements.

Expected Computational Resources

- RAM (Medium)
- CPU
  - Bursty, e.g. significant load every 10 seconds
  - Use of parallel threads desired
  - Possible use of unmanaged library to accelerate speed
- File System (Low)

Testing and Demonstration

- Algorithms will be tested for accuracy using data sets provided by developers in csv format
- Algorithms will be tested for robustness, e.g. ability to handle missing inputs, using data sets provided by developers in csv format

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Off-Line
Research-Grade

J. Synchronous Machine Parameter Estimation

T&D Consulting Engineers

MontanaTech THE UNIVERSITY OF MONTANA

Synchronous Machine Models – Justification

Text from NERC MOD-027-1: Verification of Models and Data for Turbine/Governor and Load Control or Active Power/Frequency Control Functions

R2. Each Generator Owner shall provide, for each applicable unit, a verified turbine/governor and load control or active power/frequency control model, including documentation and data (as specified in Part 2.1) to its Transmission Planner in accordance with the periodicity specified in MOD-027 Attachment 1. [Violation Risk Factor: Medium] [Time Horizon: Long-term Planning]

2.1. Each applicable unit's model shall be verified by the Generator Owner using one or more models acceptable to the Transmission Planner. Verification for individual units rated less than 20 MVA (gross nameplate rating) in a generating plant (per Section 4.2.1.2, 4.2.2.2, or 4.2.3.2) may be performed using either individual unit or aggregate unit model(s) or both. Each verification shall include the following:

2.1.1. Documentation comparing the applicable unit's MW model response to the recorded MW response for either:

- A frequency excursion from a system disturbance that meets MOD-027 Attachment 1 Note 1 with the applicable unit on-line,
- A speed governor reference change with the applicable unit online, or
- A partial load rejection test, [2]

Figure from NERC white paper "Power System Model Validation", December 2010

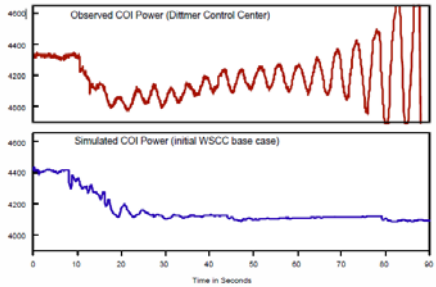


Figure 3.1 — Comparison of Observed System Response During August 10, 1996 Disturbance with Response Using Models Then in Use.

- NERC considers accurate models essential to reliability (MOD standards)
- Periodic testing required

Synchronous Machine Models – Implementation

- Current Implementation
  - Load rejection test
- Synchrophasor Enhancements
  1. openECA analytic retains a copy of the model
  2. openECA analytic waits for "significant" system event, i.e. one in which there is a measurable frequency excursion
  3. After the system event, analytic "replays" conditions into the model and reports on the accuracy of the model's response to event

DESIGN DOCUMENT

open ECA ANALYTICS DESIGN

Off-Line
Research-Grade

K. Acceleration Trend Relay Improvement

T&D Consulting Engineers

MontanaTech THE UNIVERSITY OF MONTANA

Acceleration Trend Relay – Justification

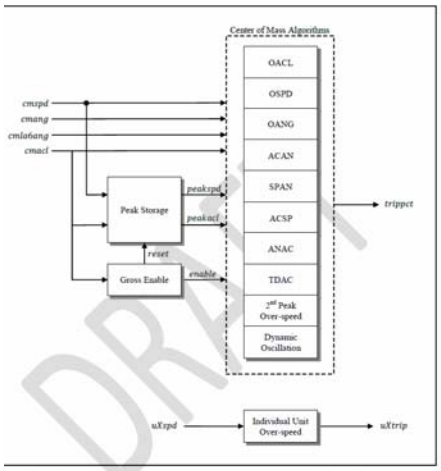
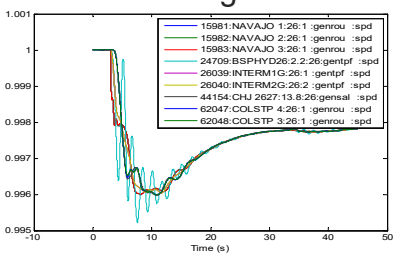


Figure 4: Trip Logic

Figure from NWE report to WECC "Acceleration Trend Relay Model Specification", November 2014.

- Protects generator from damage due to loss of synchronism
- Required in special cases where power plants are located at the end of a long line



Acceleration Trend Relay – Implementation

- Current Implementation
  - Inputs: MW output from generator; generator shaft speed measured by “toothed wheel”
    - (Acceleration is the derivative of speed.)
  - Both are local measurements
- Synchrophasor Enhancements
  - Additional input(s): Speed of remote generators
  - Algorithm enhancements: Use of relative acceleration to augment local acceleration calculations

DESIGN DOCUMENT

open ECA

10. APPENDICES

DESIGN DOCUMENT

Appendices

A. Definitions and Acronyms

A. Definitions and Acronyms

- **Components** – The elements of the openECA system that are used to realize the design functions. Components are intended to provide the necessary support for the design functions. Components can leverage many software modules in implementation.
- **CSV** – Comma-separated values, columnar data in a text format in which each column is delimited by a comma.
- **Data Conditioning** – For openECA, creating intelligently modified versions of measured phasor signals through model-less techniques (smoothing) and/or model-based techniques (linear state estimation), and creating additional phasor signals representative unmeasured network elements for increased observability.
- **GUID** – A unique reference number used as an identifier in computer software. The value of a GUID is usually represented as a 32-character hexadecimal string (e.g., 21EC2020-3AEA-1069-A2DD-08002B30309D) and stored as a 128-bit integer.
- **IEC 61850** – A standard for the design of electrical substation automation, mainly of interest for the protocols that it specifies, such as IEC 61850-90-5.
- **IEEE C37.118** – A standard used for measurement, the method of quantifying measurements, test and certification requirements for verifying accuracy, and the data transmission format and protocol for real-time communication of phasor data. Data are usually streamed in this format over UDP/IP or across a serial link.
- **Measurement** – The value of a point at a specific time, and associated per-measurement metadata.
- **Measurement Stream** – A sequence of measurements for a specific Point ID.
- **Message** – A data object that has semantic meaning to openECA and/or the CAI. It may take the form of a command or a notification. In both cases, the message is a fully encapsulated communication between modules.

A. Definitions and Acronyms

- **PDC (Phasor Data Concentrator)** – A system that consumes phasor measurement data from many PMUs, waits for a specified period to allow data to be received, and then concentrates the phasor measurements that have GPS timestamps that are in the same time window into a common output IEEE C37.118 frame.
- **Phasor** – In physics and engineering, a *phase vector*, or *phasor*, is a representation of a sine wave whose amplitude (A) and angular frequency (ω) are time-invariant. It is a subset of a more general concept called *analytic representation*. Phasors decompose the behavior of a sinusoid into three independent factors that relay amplitude, frequency, and phase information.
- **PMU (Phasor Measurement Unit)** – Any device that measures and transmits phase angle and associated measurements. For the architectures under consideration, this is usually a dedicated unit transmitting IEEE C37.118 packets.
- **Point** – A measurement point in the system; e.g., voltage at substation X bus N line 1, or current at line 12.
- **TLS (Transport Layer Security)** – A cryptographic protocol, which provides communication security over the Internet, that encrypts the segments of network connections above the transport layer using symmetric cryptography for privacy and a keyed message authentication code for message reliability. TLS is the successor to SSL (Secure Socket Layer).
- **X.509** – A standard that includes use of digital identity certificates for public key revocation and authentication of hosts.

DESIGN DOCUMENT

open ECA Appendices

B. Common Analytics Interface API Overview

B. CAI Server API Target Use Case

- Server API use is targeted for vendors that wish to implement a data provider for extremely tight integration with the openECA platform – ideal source platforms will include those that normally deal with time-series data management
- For most common cases using the Data Service, the common Server API host, will be sufficient as this tool will already have adapter based extensibility
- The Server API itself will be usable for by developers and core functionality be available for extensible use in .NET based applications (including Mono based deployments)
- Note that deployments on non-.NET based platforms will be possible but base code for Server API targeting other platforms will not be provided through this phase of the openECA project

B. CAI Server API Summary

- Base transport layer is based on ZeroMQ
  - Security mode is optional depending on deployment
- Serialization of measurements will be based on the Gateway Exchange Protocol
- Primary functions include
  - Manage client subscribes / unsubscribes with measurements specifications and filter expressions
  - Handle meta-data publication to clients for measurements and user-defined data structures
  - Serialization and persistence of user-defined data structures – must include measurement mapping

B. CAI Client API Target Use Case

- The primary function of the Client API is subscribe to streaming data being sent from one or more Server API instances and fully manage user defined data structures
- The Client API is normally accessed by analytic developers as a "function" of normal algorithm development through use of the Data Modeling Manager tool and provided analytic templates, as a result, the analytic developer rarely *directly* interacts with the API at a code level since the provided templates will already stub-in code for needed Client API interactions and functionality
- The actual client API is a set of code requirements for allowing openECA to be integrated with other development platforms related to data transport and serialization
- The API itself will be implemented on a target platform by a developer and should be based on a implementation that is similar to platform being extended, for example, if a Java version of the Client API was being developed, a set of good parallel references would be the existing C# and/or C++ implementations of the Client API

B. CAI Client API Summary

- Base transport layer is based on ZeroMQ
  - Security mode will need to match that of Server API configuration, including possible key exchange as needed
- Deserialization of measurements will be based on the Gateway Exchange Protocol
- Primary functions include
  - Subscribe / Unsubscribe to Server API hosts with measurements specifications and filter expressions
  - Handle meta-data reception from Server API hosts that contain measurements and user-defined data structures
  - Deserialization of user-defined data structures – including measurement mapping

DESIGN DOCUMENT

open ECA Appendices

C. Prerequisites and Installation for Test Harness

Tools needed to install and use the openECA prototype Data Modeling Manager and C# based analytic template, i.e., the "Test Harness" application to be used for algorithm development.

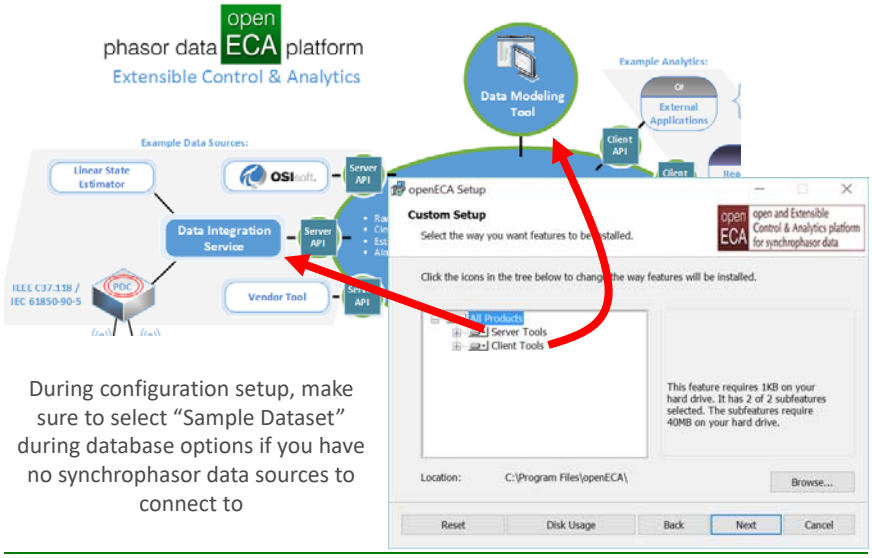
C. Development Environment

- For this prototype only C# is supported, so Visual Studio 2015 is recommended for primary analytic development and testing – the free Community Edition is fine
- Web based UI components target HTML 5 compatible browsers. The prototype has been developed using Google Chrome, so this browser is recommended, however, newer versions of Internet Explorer or Firefox may work fine, but these options have not been exhaustively tested

C. Installation Steps

- Download openECA Prototype with Analytic Development Test Harness from:
  - https://github.com/GridProtectionAlliance/openECA/releases
- Extract downloaded zip file contents into their own folder and run Setup.exe
- Note that Windows "Smart Screen" may show message that this application downloaded from the Internet is not trusted, you will need to select "Run Anyway"

C. Installation Options



During configuration setup, make sure to select "Sample Dataset" during database options if you have no synchrophasor data sources to connect to

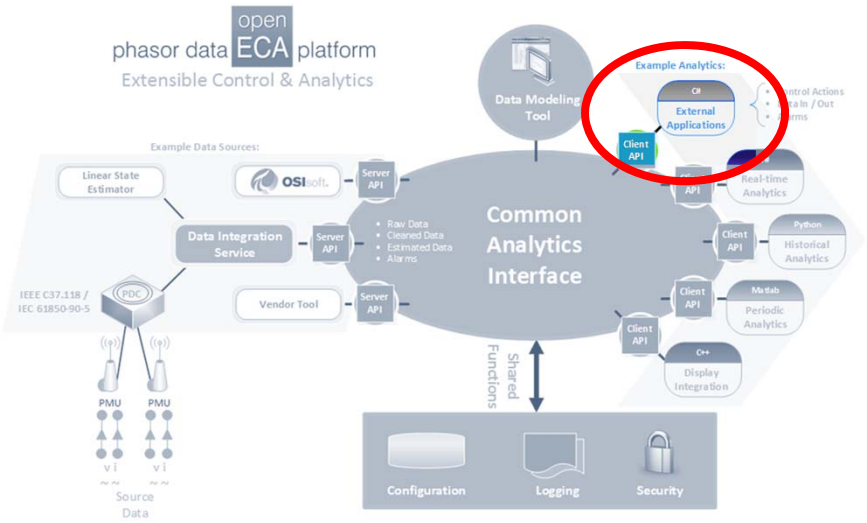
DESIGN DOCUMENT

open ECA

Appendices

D. Test Harness Use Case Example

D. Analytic Development Test Harness



D. Creating a New Analytic

- Development of a new analytic will begin with the openECA openECA Data Modeling Manager
- The analytic developer will begin by using the tool to make a connection to one or more Server API instances and start reviewing the available data sources and associated measured values
- The manager will be used to create data structures and mappings to streaming data sources, then the developer will select a target language for their analytic and use the tool to create a new “Test Harness” project

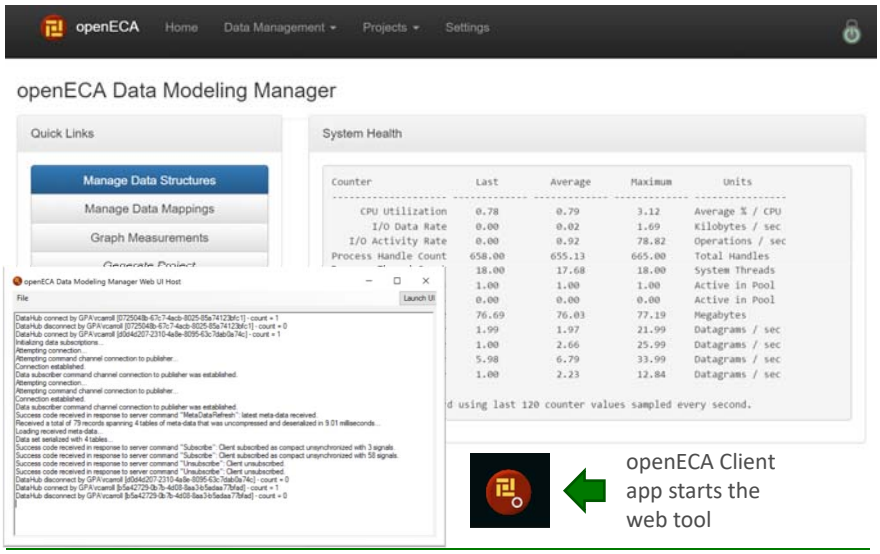
D. Use Case Objective: Calculate Power

- We will walk through an example to calculate power, i.e., Active, Reactive and Apparent power, or, MW (P), MVar (Q) and MVA (S)
- For example, the equation to calculate MW is as follows:
  - 3 * voltageMagnitude * currentMagnitude * Math.Cos(voltageAngle - currentAngle) / 1.0e+6D;

D. Analytic Data Structure Definition

- The Data Modeling Manager allows the analytic developer to define classes of data structures, including the primary input and output:
  - We call the first domain input ***SourceData****
  - The second is the analytic product called ***ResultData****
- The contents of these data structures is under the complete control of the analytic developer
- The *SourceData* will be automatically populated with time-aligned incoming data over the desired interval, in this case, 1/30 of second
  - Auto-generated code that populates the *SourceData* structure is available for the analytic writer to review and validate
- The *ResultData* will be populated by the analytic developer and is returned at the end of processing (literally the return value from the primary function)

D. Open the Data Modeling Manager Tool



openECA Client app starts the web tool

D. Define Data Types for Power Calculator

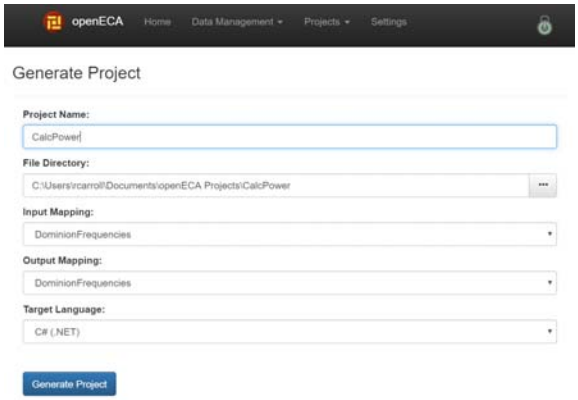
- Phasor (a complex number in polar form)
  - Angle (Double)
  - Magnitude (Double)
- VIPair (a voltage and current phasor)
  - Voltage (Phasor)
  - Current (Phasor)
- Power (MW/P, MVar/Q, and MVA/S)
  - Active (Double)
  - Reactive (Double)
  - Apparent (Double)

D. Define Mappings (Data Type "Instances")

- TestDeviceV1 (Phasor)
  - Map Angle and Magnitude
- TestDeviceI1 (Phasor)
  - Map Angle and Magnitude
- SourceData (VIPair)
  - Map Voltage to TestDeviceV1, and
  - Map Current to TestDeviceI1
- ResultData (Power)
  - Map values to PPA:1, PPA:2, PPA:3

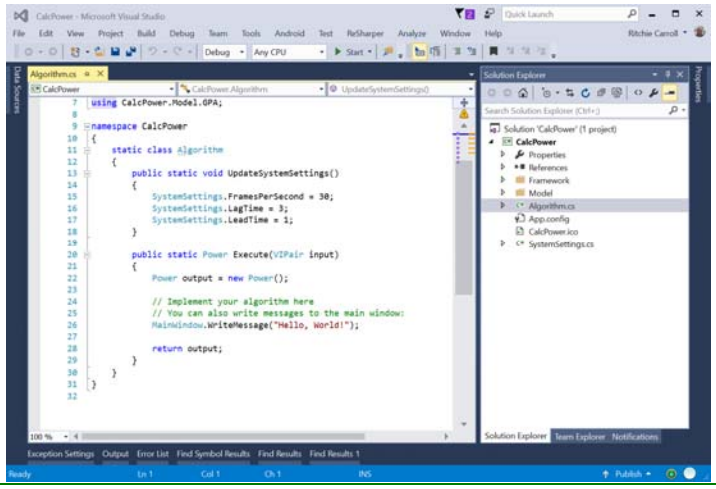
D. Generate the Project

Click “Generate Project” and name application “CalcPower”



D. Open Analytic Project in Visual Studio

…\Documents\openECA Projects\CalcPower\



D. Add Code to Analytic Entry Point

Open “Algorithm.cs” and start adding code to single function:

ResultData Mapping | SourceData Mapping

```
Power Execute(VIPair input)
{
    // Analytic code goes here…
}
```

This function will be called every 1/30 of a second with a populated “input” structure.

DESIGN DOCUMENT

open ECA

12. REFERENCES

Reference Material

- https://msdn.microsoft.com/en-us/library/ms229042(v=vs.110).aspx
- https://msdn.microsoft.com/library/ms229042%28v=vs.100%29.aspx?f=255&MSPPError=-2147217396
- https://www.youtube.com/watch?v=heh4OeB9A-c
- http://blogs.mulesoft.com/dev/api-dev/api-best-practices-series-intro/
- http://www.apiacademy.co/resources/api-design-101-api-design-basics/
- http://www4.in.tum.de/~blanchet/api-design.pdf
- https://www.redhat.com/en/files/resources/so-api-design-best-practices-red-hat-technology-whitepaper-v3.pdf
- http://www.codeproject.com/Articles/648760/Principles-of-API-Design-The-One-The-Many-The-Null
- http://www.infoq.com/presentations/effective-api-design
- http://blog.isnotworking.com/2007/05/api-design-guidelines.html