ANEXO C– Código Fonte do Aplicativo ISOSpatialQuality

Código fonte do aplicativo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class Angle
    {
        private int _deg = 0;
        private int _min = 0;
        private double _sec = 0;

        public int Deg
        {
            get { return _deg; }
            set { _deg = value; }
        }
        public int Min
        {
            get { return _min; }
            set { _min = value; }
        }
        public double Sec
        {
            get { return _sec; }
            set { _sec = value; }
        }

        public override string ToString()
        {
            string msg = Deg.ToString() + "º " + Min.ToString() + "' " + Sec.ToString();
            return msg;
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19103
{
    public class AttributeName : LocalName
    {
        private string _aName = string.Empty;
        private TypeName _attributeType = null;

        public new string aName {
            get {
                return _aName;
            }
            set {
                _aName = value;
            }
        }
        public TypeName attributeType {
            get {
                return _attributeType;
            }
            set {
                _attributeType = value;
            }
        }
    }
}
```

```
=====================================================================
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Address
    {
        private ArrayList _deliveryPoint;
        private string _city = string.Empty;
        private string _administrativeArea = string.Empty;
        private string _postalCode = string.Empty;
        private string _country = string.Empty;
        private ArrayList _electronicEmailAddress;

        public ArrayList deliveryPoint
        {
            get
            {
                return _deliveryPoint;
            }
            set
            {
                _deliveryPoint = value;
            }
        }
        public string city
        {
            get
            {
                return _city;
            }
            set
            {
                _city = value;
            }
        }
        public string administrativeArea
        {
            get
            {
                return _administrativeArea;
            }
            set
            {
                _administrativeArea = value;
            }
        }
        public string postalCode
        {
            get
            {
                return _postalCode;
            }
            set
            {
                _postalCode = value;
            }
        }
        public string country
        {
            get
            {
                return _country;
            }
            set
            {
                _country = value;
            }
        }
```

```
        public ArrayList electronicMailAddress
        {
            get
            {
                return _electronicEmailAddress;
            }
            set
            {
                _electronicEmailAddress = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Citation
    {
        private string _title = string.Empty;
        private ArrayList _alternateTitle = null;
        private CI_Date _date = new CI_Date();
        private string _edition = string.Empty;
        private DateTime _editionDate = DateTime.Now;
        private citedResponsiblePartyCollection _citedResponsibleParty = null;
        private CI_PresentationFormCode _presentationForm = CI_PresentationFormCode.undefined;
        private CI_Series _series = null;
        private string _otherCitationDetails = string.Empty;
        private string _collectiveTitle = string.Empty;
        private string _ISBN = string.Empty;
        private string _ISSN = string.Empty;
        private MD_Identifier _identifier = null;

        public MD_Identifier identifier
        {
            get
            {
                return _identifier;
            }
            set
            {
                _identifier = value;
            }
        }
        public string title
        {
            get
            {
                return _title;
            }
            set
            {
                _title = value;
            }
        }
        public ArrayList alternateTitle
        {
            get
            {
                return _alternateTitle;
            }
            set
            {
                _alternateTitle = value;
            }
        }
        public CI_Date date
        {
            get
```

```
            {
                return _date;
            }
            set
            {
                _date = value;
            }
        }
        public string edition
        {
            get
            {
                return _edition;
            }
            set
            {
                _edition = value;
            }
        }
        public DateTime editionDate
        {
            get
            {
                return _editionDate;
            }
            set
            {
                _editionDate = value;
            }
        }
        public citedResponsiblePartyCollection citedResponsibleParty
        {
            get
            {
                return _citedResponsibleParty;
            }
            set
            {
                _citedResponsibleParty = value;
            }
        }
        public CI_PresentationFormCode presentationForm
        {
            get
            {
                return _presentationForm;
            }
            set
            {
                _presentationForm = value;
            }
        }
        public CI_Series series
        {
            get
            {
                return _series;
            }
            set
            {
                _series = value;
            }
        }
        public string otherCitationDetails
        {
            get
            {
                return _otherCitationDetails;
            }
            set
            {
                _otherCitationDetails = value;
            }
        }
```

```
        public string collectiveTitle
        {
            get
            {
                return _collectiveTitle;
            }
            set
            {
                _collectiveTitle = value;
            }
        }
        public string ISBN
        {
            get
            {
                return _ISBN;
            }
            set
            {
                _ISBN = value;
            }
        }
        public string ISSN
        {
            get
            {
                return _ISSN;
            }
            set
            {
                _ISSN = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Contact
    {
        private CI_Telephone _phone = null;
        private CI_Address _address = null;
        private CI_OnlineResource _onlineResource = null;
        private string _hoursOfService = string.Empty;
        private string _contactInstructions = string.Empty;

        public CI_Telephone phone
        {
            get
            {
                return _phone;
            }
            set
            {
                _phone = value;
            }
        }
        public CI_Address address
        {
            get
            {
                return _address;
            }
            set
            {
                _address = value;
            }
        }
        public CI_OnlineResource onlineResource
```

```
        {
            get
            {
                return _onlineResource;
            }
            set
            {
                _onlineResource = value;
            }
        }
        public string hoursOfService
        {
            get
            {
                return _hoursOfService;
            }
            set
            {
                _hoursOfService = value;
            }
        }
        public string contactInstructions
        {
            get
            {
                return _contactInstructions;
            }
            set
            {
                _contactInstructions = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Date
    {
        private DateTime _date = DateTime.Now;
        private CI_DateTypeCode _dateTypeCode = CI_DateTypeCode.undefined;

        public DateTime date
        {
            get
            {
                return _date;
            }
            set
            {
                _date = value;
            }
        }
        public CI_DateTypeCode dateTypeCode
        {
            get
            {
                return _dateTypeCode;
            }
            set
            {
                _dateTypeCode = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_OnlineResource
    {
        private string _linkage = string.Empty;
        private string _protocol = string.Empty;
        private string _applicationProfile = string.Empty;
        private string _name = string.Empty;
        private string _description = string.Empty;
        private CI_OnLineFunctionCode _function = CI_OnLineFunctionCode.undefined;

        public string linkage
        {
            get { return _linkage; }
            set { _linkage = value; }
        }

        public string protocol
        {
            get { return _protocol; }
            set { _protocol = value; }
        }

        public string applicationProfile
        {
            get { return _applicationProfile; }
            set { _applicationProfile = value; }
        }

        public string name
        {
            get { return _name; }
            set { _name = value; }
        }

        public string description
        {
            get { return _description; }
            set { _description = value; }
        }

        public CI_OnLineFunctionCode function
        {
            get { return _function; }
            set { _function = value; }
        }

    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_ResponsibleParty
    {
        private string _individualName = string.Empty;
        private string _organisationName = string.Empty;
        private string _positionName = string.Empty;
        private CI_Contact _contactInfo = null;
        private CI_RoleCode _role = CI_RoleCode.undefined;

        public string individualName
        {
            get
            {
                return _individualName;
```

```
            }
            set
            {
                _individualName = value;
            }
        }
        public string organisationName
        {
            get
            {
                return _organisationName;
            }
            set
            {
                _organisationName = value;
            }
        }
        public string positionName
        {
            get
            {
                return _positionName;
            }
            set
            {
                _positionName = value;
            }
        }
        public CI_Contact contactInfo
        {
            get
            {
                return _contactInfo;
            }
            set
            {
                _contactInfo = value;
            }
        }
        public CI_RoleCode role
        {
            get
            {
                return _role;
            }
            set
            {
                _role = value;
            }
        }
    }
}

=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Series
    {
        private string _name = string.Empty;
        private string _issueIdentification = string.Empty;
        private string _page = string.Empty;

        public string name
        {
            get
            {
                return _name;
            }
            set
```

```
            {
                _name = value;
            }
        }
        public string issueIdentification
        {
            get
            {
                return _issueIdentification;
            }
            set
            {
                _issueIdentification = value;
            }
        }
        public string page
        {
            get
            {
                return _page;
            }
            set
            {
                _page = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class CI_Telephone
    {
        private ArrayList _voice = null;
        private ArrayList _facsimile = null;

        public ArrayList voice
        {
            get
            {
                return _voice;
            }
            set
            {
                _voice = value;
            }
        }
        public ArrayList facsimile
        {
            get
            {
                return _facsimile;
            }
            set
            {
                _facsimile = value;
            }
        }
    }
}

=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
```

```
namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class citedResponsiblePartyCollection : CollectionBase
    {
        public int addCI_ResponsibleParty(CI_ResponsibleParty property)
        {
            return (List.Add(property));
        }
        public void removeCI_ResponsibleParty(CI_ResponsibleParty property)
        {
            List.Remove(property);
        }
        public int IndexOf(CI_ResponsibleParty property)
        {
            return (List.IndexOf(property));
        }
        public CI_ResponsibleParty this[int Index]
        {
            get
            {
                return (CI_ResponsibleParty)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class DQ_ConformanceResult : DQ_Result
    {
        private CI_Citation _specification = null;
        private string _explanation = string.Empty;
        private bool _pass = false;

        public DQ_ConformanceResult()
        {
            _specification = new CI_Citation();
        }

        public CI_Citation specification
        {
            get { return _specification; }
            set { _specification = value; }
        }

        public string explanation
        {
            get { return _explanation; }
            set { _explanation = value; }
        }

        public bool pass
        {
            get { return _pass; }
            set { _pass = value; }
        }
    }
}
=======================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Data;
using System.Data.OleDb;

namespace ISOSpatialQuality.Utilities
{
    public class DBUtils
    {
        private string _lastErrorMsg = string.Empty;
        private string _lastErrorCode = string.Empty;

        public string lastErrorMsg
        {
            get { return _lastErrorMsg; }
            set { _lastErrorMsg = value; }
        }
        public string lastErrorCode
        {
            get { return _lastErrorCode; }
            set { _lastErrorCode = value; }
        }

        private string dbConnString = string.Empty;
        private OleDbConnectionStringBuilder myDBConnString = new OleDbConnectionStringBuilder();

        public DataTable getItemsFromDatabase(string sql)
        {
            OleDbConnection oConn = null;
            OleDbCommand oCmd = null;
            DataTable result = null;
            try
            {
                if (!string.IsNullOrEmpty(dbConnString))
                {
                    oConn = new OleDbConnection(dbConnString);
                    oConn.Open();
                    if (oConn.State == ConnectionState.Open)
                    {
                        oCmd = new OleDbCommand(sql, oConn);
                        oCmd.CommandType = CommandType.Text;
                        OleDbDataReader oRead = oCmd.ExecuteReader();
                        if (oRead != null)
                        {
                            result = new DataTable();
                            result.Load(oRead);
                        }
                    }
                }
                return result;
            }
            catch (Exception ex)
            {
                return null;
                throw ex;
            }
            finally
            {
                if (oConn.State == ConnectionState.Open)
                    oConn.Close();
                oConn = null;
                oCmd = null;
                result = null;
            }
        }

        public Dictionary<int, string> DQ_MEASURE()
        {
            Dictionary<int, string> retorno = null;
            try {
                string sql = "select ID_MEASURE, MEASURE_DESCRIPTION from DQ_MEASURES";
                DataTable dt = getItemsFromDatabase(sql);
                if (dt != null)
                {
                    retorno = new Dictionary<int, string>();
                    foreach (DataRow drow in dt.Rows)
```

```
                    {
                        int ID_MEASURE = int.Parse(drow["ID_MEASURE"].ToString());
                        string MEASURE_DESCRIPTION = drow["MEASURE_DESCRIPTION"].ToString();
                        retorno.Add(ID_MEASURE, MEASURE_DESCRIPTION);
                    }
                }
                dt = null;
                return retorno;
            }
            catch (Exception ex) {
                return null;
                throw new Exception("Error", ex);
            }
            finally {
                retorno = null;
            }
        }

        public DBUtils()
        {
            try
            {
                _lastErrorCode = string.Empty;
                _lastErrorMsg = string.Empty;

                if (!getDBConnectionString())
                {
                    _lastErrorMsg = "Could not retrieve connection string information.";
                    _lastErrorCode = "999";
                }
            }
            catch (Exception ex)
            {
                _lastErrorMsg = ex.Message.ToString();
                _lastErrorCode = "-1";
            }
        }

        ~DBUtils()
        { }

        private bool getDBConnectionString()
        {
            try
            {
                string _MDBPath = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecuti
ngAssembly().CodeBase) + "\\Settings\\SettingsDB.accdb";
                string MDBPath = _MDBPath.Substring(6);
                myDBConnString.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" + MD
BPath + ";Persist Security Info=True";
                dbConnString = myDBConnString.ConnectionString.ToString();
                return true;
            }
            catch (Exception ex)
            {
                return false;
                throw ex;
            }
        }
    }

    public class Dictionaries
    {
        public Dictionary<string, string> buildDictionary(DataTable dTable, string keyField, string key
Value)
        {
            Dictionary<string, string> retorno = null;
            try
            {
                if (dTable != null)
                {
                    retorno = new Dictionary<string, string>();
                    foreach (DataRow dRow in dTable.Rows)
                    {
```

```
                        string key = dRow[keyField].ToString();
                        string value = dRow[keyValue].ToString();
                        retorno.Add(key, value);
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                return null;
                throw ex;
            }
            finally
            {
                retorno = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;
using System.Collections;

namespace ISOSpatialQuality.Classes
{
    public class detectionResults
    {
        private int _totalElements = 0;
        private int _detectedElements = 0;
        private System.Collections.ArrayList _detectedIDs = null;

        public int totalElements {
            get {
                return _totalElements;
            }
            set {
                _totalElements = value;
            }
        }
        public int detectedElements {
            get { return _detectedElements; }
            set { _detectedElements = value; }
        }
        public System.Collections.ArrayList detectedIDs {
            get { return _detectedIDs; }
            set { _detectedIDs = value; }
        }
        public double Percentage()
        {
            double retorno = 0;
            try {
                if (_totalElements == 0)
                {
                    retorno = 0;
                }
                else
                {
                    retorno = (_detectedElements / _totalElements) * 100;
                }
                return retorno;
            }
            catch (Exception ex) {
                return 0;
                throw new Exception("Error", ex);
            }
        }
```

```
        public double Ratio()
        {
            double retorno = 0;
            if (_totalElements == 0)
            {
                retorno = 0;
            }
            else {
                retorno = _detectedElements / _totalElements;
            }
            return retorno;
        }
        public string fraction()
        {
            string retorno = string.Empty;
            retorno = _detectedElements.ToString() + " / " + _totalElements.ToString();
            return retorno;
        }
    }

    public class detectionResultsCollection : CollectionBase
    {
        public int addDetectionResults(detectionResults property)
        {
            return (List.Add(property));
        }
        public void removeDetectionResults(detectionResults property)
        {
            List.Remove(property);
        }
        public int IndexOf(detectionResults property)
        {
            return (List.IndexOf(property));
        }
        public detectionResults this[int Index]
        {
            get
            {
                return (detectionResults)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19103;
using ISOSpatialQuality.Classes.ISO._19109;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public abstract class DQ_Element
    {
        private ArrayList _nameOfMeasure = null;
        private MD_Identifier _measureIdentification = null;
        private string _measureDescription = string.Empty;
        private DQ_EvaluationMethodTypecode _evaluationMethodType = DQ_EvaluationMethodTypecode.undefin
ed;
        private string _evaluationMethodDescription = string.Empty;
        private CI_Citation _evaluationProcedure = null;
        private ArrayList _dateTime = null;
        private DQ_ResultCollection _result = null;

        public ArrayList nameOfMeasure
        {
```

```
            get { return _nameOfMeasure; }
            set { _nameOfMeasure = value; }
        }
        public MD_Identifier measureIdentification
        {
            get { return _measureIdentification; }
            set { _measureIdentification = value; }
        }
        public string measureDescription
        {
            get { return _measureDescription; }
            set { _measureDescription = value; }
        }
        public DQ_EvaluationMethodTypecode evaluationMethodType
        {
            get { return _evaluationMethodType; }
            set { _evaluationMethodType = value; }
        }
        public string evaluationMethodDescription
        {
            get { return _evaluationMethodDescription; }
            set { _evaluationMethodDescription = value; }
        }
        public CI_Citation evaluationProcedure
        {
            get { return _evaluationProcedure; }
            set { _evaluationProcedure = value; }
        }
        public ArrayList dateTime
        {
            get { return _dateTime; }
            set { _dateTime = value; }
        }
        internal DQ_ResultCollection result
        {
            get { return _result; }
            set { _result = value; }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;

namespace ISOSpatialQuality.Classes
{
    public class FeatureDefinition
    {
        private featureSchema _attributes = null;
        private featureRepresentation _symbology = null;
        private featureType _featureType = featureType.unknown;

        public featureSchema Attributes {
            get {
                return _attributes;
            }
            set {
                _attributes = value;
            }
        }
        public featureRepresentation Symbology {
            get {
                return _symbology;
            }
            set {
                _symbology = value;
```

```
            }
        }
        public featureType FeatureType {
            get {
                return _featureType;
            }
            set {
                _featureType = value;
            }
        }

        public FeatureDefinition()
        {
            _attributes = new featureSchema();
            _symbology = new featureRepresentation();
        }
    }

    public class feature
    {
        private FeatureDefinition _featureDef = null;
        private string _featureName = string.Empty;

        public FeatureDefinition featureDefinition { get { return _featureDef; } set { _featureDef = va
lue; } }
        public string featureName { get { return _featureName; } set { _featureName = value; } }
    }

    public enum DBFieldType : int
    {
        undefined,
        integer,
        real,
        datetime,
        character,
        boolean
    }

    public enum featureType : int
    {
        unknown,
        point,
        curve,
        polygon,
        surface,
        solid,
        compositePoint,
        compositeCurve,
        compositeSurface,
        compositeSolid,
        complex
    }

    public class field
    {
        private string _name = string.Empty;
        private DBFieldType _fieldType = DBFieldType.undefined;
        private object _value = null;

        public string Name {
            get {
                return _name;
            }
            set {
                _name = value;
            }
        }
        public DBFieldType FieldType {
            get {
                return _fieldType;
            }
            set {
                _fieldType = value;
            }
```

```
        }
        public object Value {
            get {
                return _value;
            }
            set {
                _value = value;
            }
        }
    }

    public class fieldCollection : CollectionBase
    {
        public int addField(field property)
        {
            int retorno = -1;
            if (!FieldExists(property))
            {
                retorno= (List.Add(property));
            }
            return retorno;
        }
        public void removeField(field property)
        {
            List.Remove(property);
        }
        public int IndexOf(field property)
        {
            return (List.IndexOf(property));
        }
        public field this[int Index]
        {
            get
            {
                return (field)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }
        public field this[string FieldName]
        {
            get {
                int i = getFieldIndex(FieldName);
                if (i > -1)
                {
                    return this[i];
                }
                else
                {
                    return null;
                }
            }
        }

        private int getFieldIndex(string fieldName)
        {
            int retorno = -1;
            try {
                if (List.Count > 0)
                {
                    for (int i = 0; i < List.Count; i++)
                    {
                        if (fieldName.ToUpper() == this[i].Name.ToUpper())
                        {
                            retorno = i;
                            break;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
```

```
                throw new Exception("Error", ex);
            }
        }

        public bool FieldExists(field property)
        {
            bool retorno = false;
            try {
                if (List.Count > 0)
                {
                    for (int i = 0; i < List.Count; i++)
                    {
                        if (property.Name.ToUpper() == this[i].Name.ToUpper())
                            retorno = true;
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                throw new Exception("Error", ex);
            }
            finally { }
        }
    }

    public class featureSchema
    {
        private fieldCollection _fieldCollection = null;

        public fieldCollection FieldCollection { get { return _fieldCollection; } set { _fieldCollectio
n = value; } }

        public featureSchema()
        {
            _fieldCollection = new fieldCollection();
        }

        ~featureSchema()
        {
            _fieldCollection = null;
        }
    }

    public class featureRepresentation
    {
        private string _lineColor = string.Empty;
        private string _lineWeight = string.Empty;
        private string _levelName = string.Empty;
        private string _lineStyle = string.Empty;

        public string LineColor
        {
            get {
                return _lineColor;
            }
            set {
                _lineColor = value;
            }
        }
        public string LineWeight {
            get {
                return _lineWeight;
            }
            set {
                _lineWeight = value;
            }
        }
        public string LevelName {
            get {
                return _levelName;
            }
            set
            {
```

```
                _levelName = value;
            }
        }
        public string LineStyle {
            get {
                return _lineStyle;
            }
            set
            {
                _lineStyle = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class DQ_Quality
    {
        private DQ_Scope _scope = null;
        private DQ_ResultCollection _report = null;

        public DQ_Scope scope {
            get {
                return _scope;
            }
            set {
                _scope = value;
            }
        }
        public DQ_ResultCollection report {
            get { return _report; }
            set { _report = value; }
        }
    }
}
=====================================================================


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19103;
using ISOSpatialQuality.Classes.ISO._19109;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class DQ_QuantitativeResult : DQ_Result
    {
        private RecordType _valueType = null;
        private string _valueUnit = string.Empty;
        private string _errorStatistic = string.Empty;
        private Record _value = null;

        public RecordType valueType
        {
            get { return _valueType; }
            set { _valueType = value; }
        }

        public string valueUnit
        {
            get { return _valueUnit; }
            set { _valueUnit = value; }
        }
```

```
        public string errorStatistic
        {
            get { return _errorStatistic; }
            set { _errorStatistic = value; }
        }

        public Record value
        {
            get { return _value; }
            set { _value = value; }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class DQ_ResultCollection : CollectionBase
    {
        public int addResult(DQ_Result property)
        {
            if (List.Count <= 1)
            {
                return (List.Add(property));
            }
            else {
                return -1;
            }
        }
        public void removeResult(DQ_Result property)
        {
            List.Remove(property);
        }
        public int IndexOf(DQ_Result property)
        {
            return (List.IndexOf(property));
        }
        public DQ_Result this[int Index]
        {
            get
            {
                return (DQ_Result)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19103;
using ISOSpatialQuality.Classes.ISO._19109;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class DQ_Scope
    {
        private MD_ScopeCode _level = null;
        private EX_Extent _extent = null;
        private MD_ScopeDescription _levelDescription = null;
```

```
        public MD_ScopeCode level {
            get {
                return _level;
            }
            set {
                _level = value;
            }
        }
        public EX_Extent extent {
            get {
                return _extent;
            }
            set {
                _extent = value;
            }
        }
        public MD_ScopeDescription levelDescription {
            get {
                return _levelDescription;
            }
            set {
                _levelDescription = value;
            }
        }
    }
}

public enum CI_DateTypeCode : int
{
    undefined,
    creation,
    publication,
    revision
}

public enum CI_RoleCode : int
{
    undefined,
    resourceProvider,
    custodian,
    owner,
    user,
    distributor,
    originator,
    pointOfContact,
    principalInvestigator,
    processor,
    publisher,
    author
}

public enum CI_OnLineFunctionCode : int
{
    undefined,
    download,
    information,
    offlineAccess,
    order,
    search
}

public enum DQ_EvaluationMethodTypecode : int
{
    undefined,
    directInternal,
    directExternal,
    indirect
}

public enum CI_PresentationFormCode : int
{
    undefined,
    documentDigital,
    documentHardcopy,
```

```
    imageDigital,
    imageHardcopy,
    mapDigital,
    mapHardcopy,
    modelDigital,
    modelHardcopy,
    profileDigital,
    profileHardcopy,
    tableDigital,
    tableHardcopy,
    videoDigital,
    videoHardcopy
}

public enum InfoSource : int
{
    Undefined,
    Provider,
    User
};

public enum CompareOperator : int
{
    undefined,
    /// <summary>
    /// eq = Equal = "="
    /// </summary>
    eq,
    /// <summary>
    /// gt = Greater Then = ">"
    /// </summary>
    gt,
    /// <summary>
    /// ge = Greater or Equal = ">="
    /// </summary>
    ge,
    /// <summary>
    /// lt = Less Then = "<"
    /// </summary>
    lt,
    /// <summary>
    /// le = Less or Equal = "<="
    /// </summary>
    le
}

public enum AcceptReject : int
{
    undefined,
    Accepted,
    Rejected
}

public enum QualityStatus : int
{
    undefined,
    accepted,
    rejected
}

public enum FeatureType : int
{
    undefined,
    pointFeature,
    linearFeature,
    PolygonFeature
}

public enum FileFormat : int
{
    undefined,
    DGN,
    DWG,
    ShapeFile
```

```
}

public enum QualitySubelement : int
{
    undefined,
    DQ_CompletenessComission,
    DQ_CompletenessOmission,
    DQ_ConceptualConsistency,
    DQ_DomainConsistency,
    DQ_FormatConsistency,
    DQ_TopologicalConsistency,
    DQ_AbsoluteExternalPositionalAccuracy,
    DQ_GriddedDataPositionalAccuracy,
    DQ_RelativeInternalPositionalAccuracy,
    DQ_AccuracyOfATimeMeasurement,
    DQ_TemporalConsistency,
    DQ_TemporalValidity,
    DQ_ThematicClassificationCorrectness,
    DQ_NonQuantitativeAttributeAccuracy,
    DQ_QuantitativeAttributeAccuracy
}

public enum ScopeCode : int
{
    undefined,
    attribute,
    attributeType,
    collectionHardware,
    collectionSession,
    dataset,
    series,
    nonGeographicDataset,
    dimensionGroup,
    feature,
    featureType,
    propertyType,
    fieldSession,
    software,
    service,
    model,
    tile
}

public enum DomainType : int
{
    undefined,
    ValueInterval,
    ListOfValues
}

public enum CompletenessAssessment : int
{
    undefined,
    CompletenessSchema,
    CompletenessRecord
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class EX_Extent
    {
        private string _description = string.Empty;
        private Ex_GeographicExtentCollection _geographicExtent = null;

        public string description
        {
            get
            {
                return _description;
```

```
            }
            set
            {
                _description = value;
            }
        }
        public Ex_GeographicExtentCollection geographicExtent
        {
            get
            {
                return _geographicExtent;
            }
            set
            {
                _geographicExtent = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class EX_GeographicBoundingBox : EX_GeographicExtent
    {
        public Angle westBoundLongitude = null;
        public Angle eastBoundLongitude = null;
        public Angle southBoundLatitude = null;
        public Angle northBoundLatitude = null;
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public abstract class EX_GeographicExtent
    {
        public bool extentTypeCode = true;
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class Ex_GeographicExtentCollection : CollectionBase
    {
        public int addGeographicExtent(EX_GeographicExtent property)
        {
            if (List.Count <= 1)
            {
                return (List.Add(property));
            }
            else
            {
                return -1;
            }

        }

        public void removeGeographicExtent(EX_GeographicExtent property)
        {
```

```
            List.Remove(property);
        }

        public int IndexOf(EX_GeographicExtent property)
        {
            return (List.IndexOf(property));
        }

        public EX_GeographicExtent this[int Index]
        {
            get
            {
                return (EX_GeographicExtent)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }
    }
}
=================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Classes
{
    public abstract class ExpectedQ
    { }

    public class ExpectedQuality : ExpectedQ
    {
        private AcceptReject _acceptOption = AcceptReject.undefined;
        private string _valueUnit = string.Empty;
        private string _value = string.Empty;
        private CompareOperator _cOperator = CompareOperator.undefined;
        private int _measureType = -1;

        public AcceptReject AcceptOption
        {
            get { return _acceptOption; }
            set { _acceptOption = value; }
        }
        public int MeasureType
        {
            get { return _measureType; }
            set { _measureType = value; }
        }
        public CompareOperator COperator
        {
            get { return _cOperator; }
            set { _cOperator = value; }
        }
        public string Value
        {
            get { return _value; }
            set { _value = value; }
        }
        public string ValueUnit
        {
            get { return _valueUnit; }
            set { _valueUnit = value; }
        }
    }

    public class ExpectedFormatConsistencyCAD : ExpectedQ
    {
```

```
        private FileFormat _fileFormat = FileFormat.undefined;
        private AcceptReject _acceptOption = AcceptReject.undefined;

        public AcceptReject acceptOption { get { return _acceptOption; } set { _acceptOption = value; }
 }
        public FileFormat fileFormat { get { return _fileFormat; } set { _fileFormat = value; } }
    }

    public class ExpectedCompleteness : ExpectedQ
    {
        private CompletenessAssessment _ca = CompletenessAssessment.undefined;
        //public CompletenessAssessment completenessAssessment { get { return _ca; } set { _ca = value;
 } }
        public virtual CompletenessAssessment completenessAssessment { get { return _ca; } set { _ca =
value; } }
    }

    public class ExpectedCompletenessSchema : ExpectedCompleteness
    {
        private AcceptReject _acceptOption = AcceptReject.undefined;
        private CompletenessAssessment _ca = CompletenessAssessment.CompletenessSchema;
        public AcceptReject acceptOption { get { return _acceptOption; } set { _acceptOption = value; }
 }
        public override CompletenessAssessment completenessAssessment { get { return _ca; } }
    }

    public class ExpectedCompletenessRecord : ExpectedCompleteness
    {
        private AcceptReject _acceptOption = AcceptReject.undefined;
        private CompletenessAssessment _ca = CompletenessAssessment.CompletenessRecord;
        private string _valueUnit = string.Empty;
        private string _value = string.Empty;
        private CompareOperator _cOperator = CompareOperator.undefined;
        private int _measureType = -1;

        public AcceptReject acceptOption { get { return _acceptOption; } set { _acceptOption = value; }
 }
        public override CompletenessAssessment completenessAssessment
        {
            get
            {
                return _ca;
            }
        }
        public int MeasureType
        {
            get { return _measureType; }
            set { _measureType = value; }
        }
        public CompareOperator COperator
        {
            get { return _cOperator; }
            set { _cOperator = value; }
        }
        public string Value
        {
            get { return _value; }
            set { _value = value; }
        }
        public string ValueUnit
        {
            get { return _valueUnit; }
            set { _valueUnit = value; }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using BIX = Bentley.Interop.Xft;

namespace ISOSpatialQuality.Forms
{
    public partial class frmComissionOmissionRecord : Form
    {
        private featureSchema _featureSchema = null;
        private QualityReport _qr = null;

        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmComissionOmissionRecord()
        {
            InitializeComponent();
        }

        private void frmComissionOmissionRecord_Load(object sender, EventArgs e)
        {
            TTMain.ShowAlways = true;
            TTMain.SetToolTip(this.listSelectedAttributes, "The fields in this area shall not have null
 values.");
            loadFeatures(this.cboFeature);
        }

        private System.Collections.ArrayList getFeatures()
        {
            System.Collections.ArrayList retorno = null;
            BIX.IFeatureEnumerator fe = null;
            BIX.FeatureMgr fm = null;
            try
            {
                fm = new BIX.FeatureMgr();
                fe = fm.GetSessionFeaturesList();
                if (fe != null)
                {
                    retorno = new System.Collections.ArrayList();
                    while (fe.MoveNext())
                    {
                        if (!isAlreadyListed(retorno, fe.Current.Name))
                        {
                            retorno.Add(fe.Current.Name);
                        }
                        Application.DoEvents();
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
                fe = null;
                fm = null;
            }
        }

        private bool isAlreadyListed(System.Collections.ArrayList list, string value)
        {
            bool retorno = false;
            try
            {
                if (list.Count > 0)
                {
                    foreach (string s in list)
```

```
                {
                    if (s.ToUpper() == value.ToUpper())
                    {
                        retorno = true;
                        break;
                    }
                }
            }
            return retorno;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return false;
        }
    }

    private void loadFeatures(ComboBox combo)
    {
        System.Collections.ArrayList features = null;
        try
        {
            features = getFeatures();
            if (features != null)
            {
                loadComboXFTFeature(combo, features);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
        finally
        {
            features = null;
        }
    }

    private void loadComboXFTFeature(ComboBox combo, System.Collections.ArrayList feats)
    {
        try
        {
            combo.Items.Clear();
            foreach (string value in feats)
            {
                combo.Items.Add(value);
            }
            string msg = "Select a Feature...";
            combo.Items.Insert(0, msg);
            combo.SelectedIndex = 0;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void listSelectedAttributes_KeyDown(object sender, KeyEventArgs e)
    {
        ListBox lista = null;
        try {
            lista = (ListBox)sender;
            if (lista.SelectedIndex > -1)
            {
                int index = lista.SelectedIndex;
                lista.Items.RemoveAt(index);
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private bool IsFieldAlreadyThere(string fieldName, ListBox lista)
```

```
        {
            bool retorno = false;
            try {
                foreach (string s in lista.Items)
                {
                    if (s.ToUpper() == fieldName.ToUpper())
                    {
                        retorno = true;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private void cmdAddField_Click(object sender, EventArgs e)
        {
            ListBox lista = null;
            try {
                lista = this.listExistingAttributes;
                if (lista.SelectedIndex > -1)
                {
                    string fieldName = lista.Text;
                    if (!IsFieldAlreadyThere(fieldName, this.listSelectedAttributes))
                    {
                        this.listSelectedAttributes.Items.Add(fieldName);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lista = null;
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void cboFeature_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox combo = null;
            try
            {
                if (this.listExistingAttributes.Items.Count > 0)
                {
                    if (MessageBox.Show("This will erase your field mapping.  Do you want to continue?"
, "Warning", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)
                    {
                        //cleanInfo();
                    }
                }
                combo = (ComboBox)sender;
                if (combo != null)
                {
                    if (combo.SelectedIndex > 0)
                    {
                        loadAttributes(combo.Text, this.listExistingAttributes);
                        //loadAttributes(combo.Text, this.listFeatureSchema);
                    }
                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
            finally { combo = null; }
        }

        private BIX.FeatureDef getFeatureDef(string featureName)
```

```
        {
            BIX.FeatureDef retorno = null;
            BIX.FeatureMgr fm = null;
            try
            {
                fm = new BIX.FeatureMgr();
                retorno = fm.GetFeatureDefinition(featureName);
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
                fm = null;
            }
        }

        private System.Collections.ArrayList getAttributeList(BIX.FeatureDef fd)
        {
            System.Collections.ArrayList retorno = null;
            try
            {
                retorno = new System.Collections.ArrayList();
                this._featureSchema = new featureSchema();
                for (int i = 0; i < fd.PropertyCount; i++)
                {
                    BIX.PropertyDef p = fd.GetPropertyDefinition(i);
                    if (p != null)
                    {
                        retorno.Add(p.Name);
                        field Field = GetFieldFromXFTProperty(p);
                        if (Field != null)
                        {
                            this._featureSchema.FieldCollection.addField(Field);
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private void loadCollectionIntoListBox(System.Collections.ArrayList lista, ListBox list)
        {
            try
            {
                list.Items.Clear();
                foreach (string s in lista)
                {
                    list.Items.Add(s);
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadAttributes(string featureName, ListBox lista)
        {
            BIX.FeatureDef fd = null;
            System.Collections.ArrayList att = null;
```

```
            try
            {
                if (!string.IsNullOrEmpty(featureName))
                {
                    fd = getFeatureDef(featureName);
                    if (fd != null)
                    {
                        att = getAttributeList(fd);
                        loadCollectionIntoListBox(att, lista);
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally
            {
                fd = null;
                att = null;
            }
        }

        private BIX.PropertyDef getPropertyInfo(string FeatureName, string FieldName)
        {
            BIX.PropertyDef retorno = null;
            BIX.FeatureMgr fm = null;
            BIX.FeatureDef fd = null;
            try
            {
                fm = new BIX.FeatureMgr();
                fd = fm.GetFeatureDefinition(FeatureName);
                if (fd != null)
                {
                    retorno = fd.GetPropertyDefinitionByName(FieldName);
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
                fm = null;
                fd = null;
            }
        }

        private field GetFieldFromXFTProperty(BIX.PropertyDef pDef)
        {
            field retorno = null;
            try
            {
                if (pDef != null)
                {
                    retorno = new field();
                    retorno.Name = pDef.Name;
                    switch (pDef.type)
                    {
                        case BIX.PropertyDataType.propertyDataTypeBoolean:
                            retorno.FieldType = DBFieldType.boolean;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeDouble:
                            retorno.FieldType = DBFieldType.real;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeInt:
                        case BIX.PropertyDataType.propertyDataTypeLong:
                            retorno.FieldType = DBFieldType.integer;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeString:
                            retorno.FieldType = DBFieldType.character;
```

```
                        break;
                    case BIX.PropertyDataType.propertyDataTypeUnknown:
                        retorno.FieldType = DBFieldType.undefined;
                        break;
                    case BIX.PropertyDataType.propertyDataTypeXmlFragment:
                        retorno.FieldType = DBFieldType.character;
                        break;
                    case BIX.PropertyDataType.propertyDataTypeYear2:
                    case BIX.PropertyDataType.propertyDataTypeYear4:
                    case BIX.PropertyDataType.propertyDataTypeDate2:
                    case BIX.PropertyDataType.propertyDataTypeDate4:
                    case BIX.PropertyDataType.propertyDataTypeDateTicks:
                        retorno.FieldType = DBFieldType.datetime;
                        break;
                }
            }
            return retorno;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally
        {
            retorno = null;
        }
    }

    private void cmdOk_Click(object sender, EventArgs e)
    {
        try {
            _featureSchema = new featureSchema();
            SearchParametersCompletenessRecord spcr = null;

            string featureName = this.cboFeature.Text;

            if (!string.IsNullOrEmpty(featureName))
            {
                spcr = new SearchParametersCompletenessRecord();

                spcr.FeatureName = featureName;
                foreach (string s in this.listSelectedAttributes.Items)
                {
                    field f = GetFieldFromXFTProperty(getPropertyInfo(featureName, s));
                    if (f != null)
                    {
                        _featureSchema.FieldCollection.addField(f);
                    }
                    f = null;
                }
            }

            if (_featureSchema.FieldCollection.Count > 0)
            {
                spcr = new SearchParametersCompletenessRecord();
                spcr.FeatureName = featureName;
                spcr.FeatureSchema = _featureSchema;
                this._qr.searchParams = spcr;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
            }

            this.Hide();
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }
  }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using BIX = Bentley.Interop.Xft;

namespace ISOSpatialQuality.Forms
{
    public partial class frmComissionOmissionSchema : Form
    {
        private featureSchema _featureSchema = null;
        private featureSchema _expectedSchema = null;
        private QualityReport _qr = null;
        private SearchParametersCompletenessSchema _spcs = null;

        public QualityReport qualityReport
        {
            get
            {
                return _qr;
            }
            set
            {
                _qr = value;
                if (_qr != null)
                {
                    if (_qr.searchParams != null)
                    {
                        _spcs = (SearchParametersCompletenessSchema)_qr.searchParams;
                    }
                    else
                    {
                        _spcs = new SearchParametersCompletenessSchema();
                    }
                }
            }
        }

        public frmComissionOmissionSchema()
        {
            InitializeComponent();
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void cmdAddField_Click(object sender, EventArgs e)
        {
            field Field = null;
            frmDefineField defineField = null;
            try {
                defineField = new frmDefineField();
                if (defineField.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    Field = defineField.Field;
                    if (Field != null)
                    {
                        //_featureSchema.FieldCollection.addField(Field);
                        _expectedSchema.FieldCollection.addField(Field);
                        this.listExpectedSchema.Items.Add(Field.Name);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
```

```
            }
            finally {
                Field = null;
                defineField.Close();
                defineField.Dispose();
            }
        }

        private void frmComissionOmissionSchema_Load(object sender, EventArgs e)
        {
            loadFeatures(this.cboFeature);
            _featureSchema = new featureSchema();
            _expectedSchema = new featureSchema();
            loadCurrentInfo();
        }

        private void loadCurrentInfo()
        {
            try {
                if (this._spcs != null)
                {
                    if (!string.IsNullOrEmpty(this._spcs.featureName.ToString()))
                        this.cboFeature.Text = this._spcs.featureName;
                    loadExpectedFeatures(this._spcs.ExpectedFeatureSchema);
                    //loadFieldMapping(this._spcs.ExpectedFeatureSchema, this._spcs.ExistingFeatureSche
ma);
                    loadFieldMapping(this._spcs);

                    //this._spcs.ExpectedFeatureSchema = null;
                    //this._spcs.ExistingFeatureSchema = null;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadFieldMapping(SearchParametersCompletenessSchema x)
        {
            try {
                if (x != null)
                {
                    if (x.FieldMapping != null)
                    {
                        System.Collections.Generic.Dictionary<field, field>.Enumerator myEnum = x.Field
Mapping.GetEnumerator();
                        while (myEnum.MoveNext())
                        {
                            field ef = myEnum.Current.Key;
                            field fa = myEnum.Current.Value;
                            string msg = ef.Name + "<==>" + fa.Name;
                            this.listMappedFields.Items.Add(msg);
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadFieldMapping(featureSchema ef, featureSchema fp)
        {
            try {
                if (ef != null)
                {
                    if (fp != null)
                    {
                        if (ef.FieldCollection.Count == fp.FieldCollection.Count)
                        {
                            this.listMappedFields.Items.Clear();
                            for (int i = 0; i < ef.FieldCollection.Count; i++)
                            {
                                string msg = ef.FieldCollection[i].Name + "<==>" + fp.FieldCollection[i
```

```
].Name;
                                this.listMappedFields.Items.Add(msg);
                            }
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadExpectedFeatures(featureSchema fs)
        {
            try {
                if (fs != null)
                {
                    if (this._expectedSchema == null)
                        this._expectedSchema = new featureSchema();
                    this.listExpectedSchema.Items.Clear();
                    foreach (field f in fs.FieldCollection)
                    {
                        this.listExpectedSchema.Items.Add(f.Name.ToString());
                        this._expectedSchema.FieldCollection.addField(f);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void updateFieldList(ListBox lista)
        {
            try {
                if (_featureSchema != null)
                {
                    lista.Items.Clear();
                    foreach (field Field in _featureSchema.FieldCollection)
                    {
                        lista.Items.Add(Field.Name);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void listExpectedSchema_KeyDown(object sender, KeyEventArgs e)
        {
            ListBox lista = null;
            try {
                if (e.KeyCode == Keys.Delete)
                {
                    lista = (ListBox)sender;
                    if (lista != null)
                    {
                        if (lista.SelectedIndex > -1)
                        {
                            //_featureSchema.FieldCollection.RemoveAt(lista.SelectedIndex);
                            _expectedSchema.FieldCollection.RemoveAt(lista.SelectedIndex);
                            lista.Items.RemoveAt(lista.SelectedIndex);
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private System.Collections.ArrayList getFeatures()
        {
```

```
            System.Collections.ArrayList retorno = null;
            BIX.IFeatureEnumerator fe = null;
            BIX.FeatureMgr fm = null;
            try
            {
                fm = new BIX.FeatureMgr();
                fe = fm.GetSessionFeaturesList();
                if (fe != null)
                {
                    retorno = new System.Collections.ArrayList();
                    while (fe.MoveNext())
                    {
                        if (!isAlreadyListed(retorno, fe.Current.Name))
                        {
                            retorno.Add(fe.Current.Name);
                        }
                        Application.DoEvents();
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
                fe = null;
                fm = null;
            }
        }

        private bool isAlreadyListed(System.Collections.ArrayList list, string value)
        {
            bool retorno = false;
            try
            {
                if (list.Count > 0)
                {
                    foreach (string s in list)
                    {
                        if (s.ToUpper() == value.ToUpper())
                        {
                            retorno = true;
                            break;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private void loadFeatures(ComboBox combo)
        {
            System.Collections.ArrayList features = null;
            try {
                features = getFeatures();
                if (features != null)
                {
                    loadComboXFTFeature(combo, features);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                features = null;
```

```
            }
        }

        private void loadComboXFTFeature(ComboBox combo, System.Collections.ArrayList feats)
        {
            try
            {
                combo.Items.Clear();
                foreach (string value in feats)
                {
                    combo.Items.Add(value);
                }
                string msg = "Select a Feature...";
                combo.Items.Insert(0, msg);
                combo.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cboFeature_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox combo = null;
            try
            {
                if (this.listMappedFields.Items.Count > 0)
                {
                    if (MessageBox.Show("This will erase your field mapping.  Do you want to continue?"
, "Warning", MessageBoxButtons.YesNo) == System.Windows.Forms.DialogResult.Yes)
                    {
                        cleanInfo();
                    }
                }
                combo = (ComboBox)sender;
                if (combo != null)
                {
                    if (combo.SelectedIndex > 0)
                    {
                        loadAttributes(combo.Text, this.listFeatureSchema);
                    }
                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
            finally { combo = null; }
        }

        private void cleanInfo()
        {
            try {
                if (this._spcs != null)
                {
                    this._spcs.ExistingFeatureSchema.FieldCollection.Clear();
                    this._spcs.ExpectedFeatureSchema.FieldCollection.Clear();
                    this._spcs.FieldMapping.Clear();
                    this.listMappedFields.Items.Clear();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadAttributes(string featureName, ListBox lista)
        {
            BIX.FeatureDef fd = null;
            System.Collections.ArrayList att = null;
            try {
                if (!string.IsNullOrEmpty(featureName))
                {
                    fd = getFeatureDef(featureName);
                    if (fd != null)
                    {
```

```
                    att = getAttributeList(fd);
                    loadCollectionIntoListBox(att, lista);
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
        finally {
            fd = null;
            att = null;
        }
    }

    private BIX.FeatureDef getFeatureDef(string featureName)
    {
        BIX.FeatureDef retorno = null;
        BIX.FeatureMgr fm = null;
        try {
            fm = new BIX.FeatureMgr();
            retorno = fm.GetFeatureDefinition(featureName);
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally {
            retorno = null;
            fm = null;
        }
    }

    private System.Collections.ArrayList getAttributeList(BIX.FeatureDef fd)
    {
        System.Collections.ArrayList retorno = null;
        try
        {
            retorno = new System.Collections.ArrayList();
            for (int i = 0; i < fd.PropertyCount; i++)
            {
                BIX.PropertyDef p = fd.GetPropertyDefinition(i);
                if (p != null)
                {
                    retorno.Add(p.Name);
                    field Field = GetFieldFromXFTProperty(p);
                    if (Field != null)
                    {
                        this._featureSchema.FieldCollection.addField(Field);
                    }
                }
            }
            return retorno;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally
        {
            retorno = null;
        }
    }

    private void loadCollectionIntoListBox(System.Collections.ArrayList lista, ListBox list)
    {
        try
        {
            list.Items.Clear();
            foreach (string s in lista)
            {
                list.Items.Add(s);
            }
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            if (this.cboFeature.SelectedIndex > 0)
            {
                this._spcs.featureName = this.cboFeature.Text;
                this._spcs.ExistingFeatureSchema = this._featureSchema;
                this._spcs.ExpectedFeatureSchema = this._expectedSchema;
                this._qr.searchParams = this._spcs;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Hide();
            }
            else
            {
                this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
                this.Hide();
            }
        }

        private BIX.PropertyDef getPropertyInfo(string FeatureName, string FieldName)
        {
            BIX.PropertyDef retorno = null;
            BIX.FeatureMgr fm = null;
            BIX.FeatureDef fd = null;
            try {
                fm = new BIX.FeatureMgr();
                fd = fm.GetFeatureDefinition(FeatureName);
                if (fd != null)
                {
                    retorno = fd.GetPropertyDefinitionByName(FieldName);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
                fm = null;
                fd = null;
            }
        }

        private field GetFieldFromXFTProperty(BIX.PropertyDef pDef)
        {
            field retorno = null;
            try {
                if (pDef != null)
                {
                    retorno = new field();
                    retorno.Name = pDef.Name;
                    switch (pDef.type)
                    {
                        case BIX.PropertyDataType.propertyDataTypeBoolean:
                            retorno.FieldType = DBFieldType.boolean;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeDouble:
                            retorno.FieldType = DBFieldType.real;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeInt:
                        case BIX.PropertyDataType.propertyDataTypeLong:
                            retorno.FieldType = DBFieldType.integer;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeString:
                            retorno.FieldType = DBFieldType.character;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeUnknown:
```

```
                            retorno.FieldType = DBFieldType.undefined;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeXmlFragment:
                            retorno.FieldType = DBFieldType.character;
                            break;
                        case BIX.PropertyDataType.propertyDataTypeYear2:
                        case BIX.PropertyDataType.propertyDataTypeYear4:
                        case BIX.PropertyDataType.propertyDataTypeDate2:
                        case BIX.PropertyDataType.propertyDataTypeDate4:
                        case BIX.PropertyDataType.propertyDataTypeDateTicks:
                            retorno.FieldType = DBFieldType.datetime;
                            break;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private field getFieldFromXFTFeature(string featureName, string fieldName)
        {
            field retorno = null;
            BIX.PropertyDef pd = null;
            try {
                pd = getPropertyInfo(featureName, fieldName);
                if (pd != null)
                {
                    retorno = GetFieldFromXFTProperty(pd);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
                pd = null;
            }
        }

        private field getFeatureField(ListBox featurePropertylist, string FeatureName, string PropertyN
ame)
        {
            field retorno = null;
            try {
                if (!string.IsNullOrEmpty(FeatureName))
                {
                    if (!string.IsNullOrEmpty(PropertyName))
                    {
                        if (featurePropertylist.SelectedIndex > -1)
                        {
                            retorno = getFieldFromXFTFeature(FeatureName, PropertyName);
                        }
                    }
                }
                return retorno;
            }
            catch(Exception ex) {
                MessageBox.Show(ex.Message.ToString(),"Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private field getFeatureField(ListBox featurePropertyList)
```

```
        {
            field retorno = null;
            try {
                if (featurePropertyList.SelectedIndex > -1)
                {
                    string fieldName = featurePropertyList.Text;
                    retorno = _featureSchema.FieldCollection[fieldName];
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private field getExpectedField(ListBox expectedPropertylist)
        {
            field retorno = null;
            try {
                if (expectedPropertylist.SelectedIndex > -1)
                {
                    string fieldname = expectedPropertylist.Text;
                    retorno = _expectedSchema.FieldCollection[fieldname];
                }
                return retorno;
            }
            catch(Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool AddFieldMapping(field expectedField, field featureField, ListBox MapList)
        {
            bool retorno = false;
            try {
                if (featureField != null)
                {
                    if (expectedField != null)
                    {
                        //this._spcs.ExistingFeatureSchema.FieldCollection.addField(featureField);
                        //this._spcs.ExpectedFeatureSchema.FieldCollection.addField(expectedField);
                        this._spcs.FieldMapping.Add(expectedField, featureField);

                        string msg = expectedField.Name + "<==>" + featureField.Name;
                        MapList.Items.Add(msg);
                        retorno = true;
                    }
                }

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private void PrepareFieldMapping(string featureFieldName, string featureName)
        {
            field featureField = null;
            field expectedField = null;
            try {
                //featureField = getFeatureField(this.listFeatureSchema, featureName, featureFieldName)
;
```

```
                featureField = getFeatureField(this.listFeatureSchema);
                expectedField = getExpectedField(this.listExpectedSchema);
                AddFieldMapping(expectedField, featureField, this.listMappedFields);
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                featureField = null;
                expectedField = null;
            }
        }

        private void cmdMapFields_Click(object sender, EventArgs e)
        {
            //field featureField = null;
            //field expectedField = null;
            try {
                string fieldName = listFeatureSchema.Text;
                string featureName = this.cboFeature.Text;
                PrepareFieldMapping(fieldName, featureName);

            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                //featureField = null;
                //expectedField = null;
            }
        }

        private void listMappedFields_KeyDown(object sender, KeyEventArgs e)
        {
            ListBox lista = null;
            try {
                lista = (ListBox)sender;
                if (lista.SelectedIndex > -1)
                {
                    if (e.KeyCode == Keys.Delete)
                    {
                        field keyfield = null;
                        int index = lista.SelectedIndex;

                        string txt = lista.Text;
                        int mIndex = txt.IndexOf("<==>");
                        string keyfeaturename = txt.Substring(0, mIndex);



                        //keyfield = this._spcs.ExpectedFeatureSchema.FieldCollection[index];
                        //keyfield = this._expectedSchema.FieldCollection[index];
                        keyfield = this._expectedSchema.FieldCollection[keyfeaturename];

                        //this._spcs.ExistingFeatureSchema.FieldCollection.RemoveAt(index);
                        //this._spcs.ExpectedFeatureSchema.FieldCollection.RemoveAt(index);
                        this._spcs.FieldMapping.Remove(keyfield);
                        lista.Items.RemoveAt(index);

                        keyfield = null;
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lista = null;
            }
        }
    }
}
=================================================================
using System;
```

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmCompletenessChoice : Form
    {
        private CompletenessAssessment _compAssess = CompletenessAssessment.undefined;
        private QualityReport _qr = null;

        public CompletenessAssessment completenessAssessment { get { return _compAssess; } set { _compA
ssess = value; } }
        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmCompletenessChoice()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private ExpectedCompleteness setQualityExpectation(CompletenessAssessment ca, QualityReport qr)
        {
            ExpectedCompleteness retorno = null;
            try {
                if (qr != null)
                {
                    if (qr.expectedQuality != null)
                    {
                        if (qr.expectedQuality.GetType() == typeof(ExpectedCompletenessSchema))
                        {
                            ExpectedCompletenessSchema ecs = (ExpectedCompletenessSchema)qr.expectedQua
lity;
                            retorno = ecs;
                        }
                        if (qr.expectedQuality.GetType() == typeof(ExpectedCompletenessRecord))
                        {
                            ExpectedCompletenessRecord ecr = (ExpectedCompletenessRecord)qr.expectedQua
lity;
                            retorno = ecr;
                        }
                    }
                    else
                    {
                        if (ca == CompletenessAssessment.CompletenessRecord)
                        {
                            //to-do
                            ExpectedCompletenessRecord ecr = new ExpectedCompletenessRecord();
                            ecr.acceptOption = AcceptReject.undefined;
                            retorno = ecr;
                            //to-do
                        }
                        if (ca == CompletenessAssessment.CompletenessSchema)
                        {
                            ExpectedCompletenessSchema ecs = new ExpectedCompletenessSchema();
                            ecs.acceptOption = AcceptReject.undefined;
                            retorno = ecs;
                        }
                    }
                }
```

```
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally {
            retorno = null;
        }
    }

    private void cmdOk_Click(object sender, EventArgs e)
    {
        _compAssess = (CompletenessAssessment)this.cboCompletenessChoice.SelectedIndex;
        ExpectedCompleteness ec = null;

        ec = setQualityExpectation(_compAssess, this._qr);

        if (this._qr.expectedQuality != null)
        {
            if (_compAssess == CompletenessAssessment.CompletenessSchema)
            {
                if (this._qr.expectedQuality.GetType() != typeof(ExpectedCompletenessSchema))
                {
                    this._qr.expectedQuality = ec;
                }
            }
            if (_compAssess == CompletenessAssessment.CompletenessRecord)
            {
                if (this._qr.expectedQuality.GetType() != typeof(ExpectedCompletenessRecord))
                {
                    this._qr.expectedQuality = ec;
                }
            }
            if (_compAssess == CompletenessAssessment.undefined)
            {
                this._qr.expectedQuality = ec;
            }
        }
        else
        {
            this._qr.expectedQuality = ec;
        }

        this.DialogResult = System.Windows.Forms.DialogResult.OK;
        this.Hide();
    }

    private void loadComboOption(ComboBox combo)
    {
        try {
            combo.DataSource = Enum.GetNames(typeof(CompletenessAssessment));
            combo.SelectedIndex=0;
        }
        catch(Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Warning");
        }
    }

    private void frmCompletenessChoice_Load(object sender, EventArgs e)
    {
        this.StartPosition = FormStartPosition.CenterScreen;
        loadComboOption(this.cboCompletenessChoice);
        loadInfo();
    }

    private void loadInfo()
    {
        try {
            if (_qr != null)
            {
                if (_qr.expectedQuality != null)
                {
                    if (_qr.expectedQuality.GetType() == typeof(ExpectedCompletenessSchema))
```

```
                        {
                            this._compAssess = CompletenessAssessment.CompletenessSchema;
                        }
                        else if (_qr.expectedQuality.GetType() == typeof(ExpectedCompletenessRecord))
                        {
                            this._compAssess = CompletenessAssessment.CompletenessRecord;
                        }
                        else {
                            this._compAssess = CompletenessAssessment.undefined;
                        }

                        //this.cboCompletenessChoice.SelectedIndex = ec.completenessAssessment.GetHashC
ode();
                        this.cboCompletenessChoice.SelectedIndex = this._compAssess.GetHashCode();
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cboCompletenessChoice_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox combo = null;
            try {
                combo = (ComboBox)sender;
                CompletenessAssessment ca = (CompletenessAssessment)combo.SelectedIndex;
                string msg = string.Empty;
                switch (ca)
                {
                    case CompletenessAssessment.undefined:
                        msg = string.Empty;
                        break;
                    case CompletenessAssessment.CompletenessSchema:
                        msg = "Check comission/omission at schema level.  For example, if the schema is
 missing attributes or, on the other hand, have excess of attributes.";
                        break;
                    case CompletenessAssessment.CompletenessRecord:
                        msg = "Check comission/omission at record level.  For example, if the mandatory
 records related to a feature have null values.";
                        break;
                    default:
                        msg = string.Empty;
                        break;
                }
                this.lblCompletenessDescription.Text = msg;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                combo = null;
            }
        }
    }
}
=====================================================================
Using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
```

```
    public partial class frmCompletenessSchemaAcceptanceCriteriaForm : Form
    {
        //private ExpectedCompleteness ec = null;
        private ExpectedCompletenessSchema ec = null;
        private QualityReport _qr = null;
        private AcceptReject ar = AcceptReject.undefined;

        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmCompletenessSchemaAcceptanceCriteriaForm()
        {
            InitializeComponent();
        }

        private void frmCompletenessSchemaAcceptanceCriteriaForm_ResizeEnd(object sender, EventArgs e)
        {
            this.Width=318;
            this.Height=150;
        }

        private void frmCompletenessSchemaAcceptanceCriteriaForm_Load(object sender, EventArgs e)
        {
            //ec = new ExpectedCompleteness();
            ec = new ExpectedCompletenessSchema();
            this.cboPassFail.SelectedIndex = 0;
            //loadInfo();
            LoadQualityInfo();
        }

        private void LoadQualityInfo()
        {
            try {
                if (this._qr != null)
                {
                    if (this._qr.expectedQuality != null)
                    {
                        if (this._qr.expectedQuality.GetType() == typeof(ExpectedCompletenessSchema))
                        {
                            ec = (ExpectedCompletenessSchema)this._qr.expectedQuality;
                            if (ec != null)
                            {
                                this.cboPassFail.SelectedIndex = ec.acceptOption.GetHashCode();
                            }
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                if (cboPassFail.SelectedIndex > -1)
                {
                    ar = (AcceptReject)this.cboPassFail.SelectedIndex;
                }
                ec.acceptOption = ar;
                _qr.expectedQuality = ec;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Hide();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }
```

```
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19115;
using ISOSpatialQuality.Classes.ISO._19109;
using ISOSpatialQuality.Classes.ISO._19103;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmDataHistory : Form
    {
        private LI_History _history = null;

        public LI_History History { get { return _history; } set { _history = value; } }

        public frmDataHistory()
        {
            InitializeComponent();
        }

        private void loadInfo()
        {
            try {
                if (_history != null)
                {
                    DateTime date = _history.DateSource;
                    string info = _history.History;
                    this.dtHistoryDate.Text = date.ToShortDateString();
                    this.txtHistoryDescription.Text = info;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            string description = string.Empty;
            DateTime date = DateTime.Today;
            try {
                bool result = DateTime.TryParse(this.dtHistoryDate.Text, out date);
                if (result)
                {
                    description = this.txtHistoryDescription.Text;
                    if (_history == null)
                    {
                        _history = new LI_History();
                    }
                    _history.DateSource = date;
                    _history.History = description;
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void frmDataHistory_Load(object sender, EventArgs e)
        {
            loadInfo();
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;

namespace ISOSpatialQuality.Forms
{
    public partial class frmDefineField : Form
    {
        private field _field = null;

        public field Field { get { return _field; } set { _field = value; } }

        private void loadFieldInfo()
        {
            try {
                if (_field != null)
                {
                    this.txtFieldName.Text = _field.Name;
                    this.cboFieldType.SelectedIndex = _field.FieldType.GetHashCode();
                }
                else
                {
                    this.txtFieldName.Text = string.Empty;
                    this.cboFieldType.SelectedIndex = 0;
                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
        }

        public frmDefineField()
        {
            InitializeComponent();
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                _field = new field();
                _field.Name = this.txtFieldName.Text;
                _field.FieldType = (DBFieldType)this.cboFieldType.SelectedIndex;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Hide();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
```

```
        }

        private void loadFiedlType(ComboBox cbo)
        {
            try {
                cbo.DataSource = Enum.GetValues(typeof(DBFieldType));
                cbo.SelectedIndex = 0;
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
        }

        private void frmDefineField_Load(object sender, EventArgs e)
        {
            try {
                loadFiedlType(this.cboFieldType);
                loadFieldInfo();
                this.txtFieldName.Focus();
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
        }
    }
}
=================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Utilities;
using BIM = Bentley.Interop.MicroStationDGN;
using BIX = Bentley.Interop.Xft;

namespace ISOSpatialQuality.Forms
{
    public partial class frmExpectedFeatureSchema : Form
    {
        private QualityReport _qr = null;

        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmExpectedFeatureSchema()
        {
            InitializeComponent();
        }

        private void dList_KeyDown(object sender, KeyEventArgs e)
        {
            ListBox list = (ListBox)sender;
            try {
                if (list.SelectedIndex > -1)
                {
                    if (e.KeyCode == Keys.Delete)
                    {
                        list.Items.RemoveAt(list.SelectedIndex);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(),"Error");
            }
            finally {
                list = null;
            }
        }

        private void frmExpectedFeatureSchema_Load(object sender, EventArgs e)
        {
```

```
            System.Collections.ArrayList lista = null;
            try {
                Application.UseWaitCursor = true;
                Application.DoEvents();
                lista = getFeatures();
                loadComboXFTFeature(this.cboFeature, lista);
                this.rdListOfValues.Checked = false;
                this.rdValueInterval.Checked = true;
                this.rdValueInterval_Click(this.rdValueInterval, null);
                loadSearchInfo();
                Application.UseWaitCursor = false;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lista = null;
            }
        }

        private bool isAlreadyListed(System.Collections.ArrayList list, string value)
        {
            bool retorno = false;
            try {
                if (list.Count > 0)
                {
                    foreach (string s in list)
                    {
                        if (s.ToUpper() == value.ToUpper())
                        {
                            retorno = true;
                            break;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private System.Collections.ArrayList getFeatures()
        {
            System.Collections.ArrayList retorno = null;
            BIX.IFeatureEnumerator fe = null;
            BIX.FeatureMgr fm = null;
            try {
                fm = new BIX.FeatureMgr();
                fe = fm.GetSessionFeaturesList();
                if (fe != null)
                {
                    retorno = new System.Collections.ArrayList();
                    while (fe.MoveNext())
                    {
                        if (!isAlreadyListed(retorno, fe.Current.Name))
                        {
                            retorno.Add(fe.Current.Name);
                        }
                        Application.DoEvents();
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
                fe = null;
                fm = null;
            }
```

```
        }

        private string getUUID(string txt)
        {
            string retorno = string.Empty;
            try {
                string[] aux = txt.Split(new char[] { ';' });
                retorno = aux[1];
                return retorno;
            }
            catch (Exception ex) {
                return string.Empty;
                throw new Exception("Error", ex);
            }
        }

        private string getFeatName(string txt)
        {
            string retorno = string.Empty;
            try
            {
                string[] aux = txt.Split(new char[] { ';' });
                retorno = aux[0];
                return retorno;
            }
            catch (Exception ex)
            {
                return string.Empty;
                throw new Exception("Error", ex);
            }
        }

        private void loadComboXFTFeature(ComboBox combo, System.Collections.ArrayList feats)
        {
            try {
                combo.Items.Clear();
                foreach (string value in feats)
                {
                    combo.Items.Add(value);
                }
                string msg = "Select a Feature...";
                combo.Items.Insert(0, msg);
                combo.SelectedIndex = 0;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            SearchParametersDomainConsistency spdc = null;
            DomainValues dv = null;
            try {
                spdc = new SearchParametersDomainConsistency();
                if (this.rdValueInterval.Checked)
                {
                    dv = setIntervalValues();
                }
                else
                {
                    dv = setListOfDomain();
                }
                if (dv != null)
                {
                    spdc.DomainValues = dv;
                }
                _qr.searchParams = spdc;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Hide();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
```

```
            finally {
                spdc = null;
                dv = null;
            }
        }

        private ListOfDomainValues setListOfDomain()
        {
            ListOfDomainValues retorno = null;
            try {
                retorno = new ListOfDomainValues();
                foreach (string s in this.dList.Items)
                {
                    if (!string.IsNullOrEmpty(s))
                    {
                        retorno.ListOfValues.addValue(s);
                    }
                }
                retorno.domainType = DomainType.ListOfValues;
                retorno.featureName = this.cboFeature.Text;
                retorno.attributeName = this.cboAttr.Text;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private intervalClass setIntervalValues()
        {
            intervalClass retorno = null;
            try {
                int min = this.IntervalControl.minValue;
                int max = this.IntervalControl.maxValue;
                retorno = new intervalClass();
                retorno.min = min;
                retorno.max = max;
                retorno.domainType = DomainType.ValueInterval;
                retorno.featureName = this.cboFeature.Text;
                retorno.attributeName = this.cboAttr.Text;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void enableCommands_ListOfValues(bool value)
        {
            this.dList.Enabled = value;
            this.cmdAdd.Enabled = value;
            this.txtListValue.Enabled = value;

        }

        private void enableCommands_Interval(bool value)
        {
            this.IntervalControl.Enabled = value;
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }
```

```
        private void cboFeature_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox combo = null;
            try {
                combo = (ComboBox)sender;
                if (combo.SelectedIndex > 0)
                {
                    FillAttributeCombo(combo.Text);
                }
                else
                {
                    this.cboAttr.Items.Clear();
                    this.cboAttr.Items.Add("Select an Attribute...");
                    this.cboAttr.SelectedIndex = 0;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                combo = null;
            }
        }

        private void FillAttributeCombo(string featureName)
        {
            BIX.FeatureMgr fm = null;
            BIX.FeatureDef fd = null;
            System.Collections.ArrayList properties = null;
            try {
                fm = new BIX.FeatureMgr();
                fd = fm.GetFeatureDefinition(featureName);
                if (fd != null)
                {
                    properties = getAttributeList(fd);
                    if (properties != null)
                    {
                        loadCollectionIntoCombo(properties, this.cboAttr, "Select an Attribute...");
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                fm = null;
                fd = null;
                properties = null;
            }
        }

        private System.Collections.ArrayList getAttributeList(BIX.FeatureDef fd)
        {
            System.Collections.ArrayList retorno = null;
            try {
                retorno = new System.Collections.ArrayList();
                for (int i = 0; i < fd.PropertyCount; i++)
                {
                    BIX.PropertyDef p = fd.GetPropertyDefinition(i);
                    if (p != null)
                    {
                        retorno.Add(p.Name);
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }
```

```
        private void loadCollectionIntoCombo(System.Collections.ArrayList lista, ComboBox combo, string
StartMessage)
        {
            try {
                combo.Items.Clear();
                foreach (string s in lista)
                {
                    combo.Items.Add(s);
                }
                if (!string.IsNullOrEmpty(StartMessage))
                {
                    combo.Items.Insert(0, StartMessage);
                    combo.SelectedIndex = 0;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(),"Error");
            }
        }

        private void cmdAdd_Click(object sender, EventArgs e)
        {
            try {
                string value = this.txtListValue.Text;
                this.dList.Items.Add(value);
                this.txtListValue.Text = string.Empty;
                this.txtListValue.Focus();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void rdValueInterval_Click(object sender, EventArgs e)
        {
            RadioButton rd = null;
            try {
                rd = (RadioButton)sender;
                enableCommands_Interval(rd.Checked);
                enableCommands_ListOfValues(!rd.Checked);
                this.rdListOfValues.Checked = !rd.Checked;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                rd = null;
            }
        }

        private void rdListOfValues_Click(object sender, EventArgs e)
        {
            RadioButton rd = null;
            try
            {
                rd = (RadioButton)sender;
                enableCommands_Interval(!rd.Checked);
                enableCommands_ListOfValues(rd.Checked);
                this.rdValueInterval.Checked = !rd.Checked;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally
            {
                rd = null;
            }
        }

        private void loadIntervalValues(DomainValues dv)
        {
            intervalClass ic = null;
```

```
            try {
                if (dv != null)
                {
                    ic = (intervalClass)dv;
                    this.cboFeature.Text = ic.featureName;
                    this.cboAttr.Text = ic.attributeName;
                    this.IntervalControl.minValue = ic.min;
                    this.IntervalControl.maxValue = ic.max;
                    this.rdValueInterval.Checked = true;
                    this.rdValueInterval_Click(this.rdValueInterval, null);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                ic = null;
            }
        }

        private void loadListOfValues(DomainValues dv)
        {
            ListOfDomainValues lodv = null;
            try {
                if (dv != null)
                {
                    lodv = (ListOfDomainValues)dv;
                    this.dList.Items.Clear();
                    foreach (string s in lodv.ListOfValues)
                    {
                        this.dList.Items.Add(s);
                    }
                    this.cboFeature.Text = dv.featureName;
                    this.cboAttr.Text = dv.attributeName;
                    this.rdListOfValues.Checked = true;
                    this.rdListOfValues_Click(this.rdListOfValues, null);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lodv = null;
            }
        }

        private void loadSearchInfo()
        {
            try {
                if (_qr != null)
                {
                    SearchParametersDomainConsistency spdc = (SearchParametersDomainConsistency)_qr.sea
rchParams;
                    if (spdc != null)
                    {
                        DomainValues dv = spdc.DomainValues;
                        if (dv != null)
                        {
                            switch (dv.domainType)
                            {
                                case DomainType.undefined:
                                    break;
                                case DomainType.ListOfValues:
                                    loadListOfValues(dv);
                                    break;
                                case DomainType.ValueInterval:
                                    loadIntervalValues(dv);
                                    break;
                                default:
                                    break;
                            }
                        }
                    }
                }
```

```
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;

namespace ISOSpatialQuality.Forms
{
    public partial class frmFileFormatValidation : Form
    {
        public frmFileFormatValidation()
        {
            InitializeComponent();
        }

        private void frmFileFormatValidation_Load(object sender, EventArgs e)
        {
            loadFileTypes(this.cboFileFormat);
        }

        private void loadFileTypes(ComboBox combo)
        {
            try {
                combo.DataSource = Enum.GetValues(typeof(FileFormat));
                combo.SelectedIndex = 0;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {

        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Utilities;
using BIM = Bentley.Interop.MicroStationDGN;

namespace ISOSpatialQuality.Forms
{
    public partial class frmFormatConsistencyGeometryonly : Form
    {
```

```
        private QualityReport _qr = null;

        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmFormatConsistencyGeometryonly()
        {
            InitializeComponent();
        }

        private void setComboByText(string valor, ComboBox cbo)
        {
            try {
                cbo.SelectedIndex = cbo.FindStringExact(valor);
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(),"Error");}
        }

        private void setComboByIndex(int index, ComboBox cbo)
        {
            try {
                cbo.SelectedIndex = index;
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
        }

        private void loadSearchInfo()
        {
            SearchParametersFormatConsistency spfc = null;
            try {
                if (_qr != null)
                {
                    if (_qr.searchParams != null)
                    {
                        spfc = (SearchParametersFormatConsistency)this._qr.searchParams;
                        if (spfc != null)
                        {
                            string layerName = spfc.LayerName;
                            setComboByText(layerName, this.cboLevelname);
                            int featureT = spfc.Featuretype.GetHashCode();
                            setComboByIndex(featureT, this.cboFeatureType);
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private System.Collections.ArrayList getDGNLevels()
        {
            System.Collections.ArrayList dLevels = null;
            Bentley.Interop.MicroStationDGN.Application dMstn;
            try
            {
                dMstn = ISOSpatialQuality.ComApp;

                if (dMstn.ActiveDesignFile.Levels.Count > 0)
                {
                    dMstn = ISOSpatialQuality.ComApp;

                    dLevels = new System.Collections.ArrayList();

                    foreach (BIM.Level dLevel in dMstn.ActiveDesignFile.Levels)
                    {
                        string LevelName = dLevel.Name.ToString();
                        dLevels.Add(LevelName);
                    }
                }

                return dLevels;
            }
            catch (Exception ex)
            {
```

```
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                dMstn = null;
                dLevels = null;
            }
        }

        private void loadLevelsinComboBox(System.Collections.ArrayList levels, ComboBox dCombo)
        {
            try
            {
                dCombo.Items.Clear();
                foreach (string lName in levels)
                {
                    dCombo.Items.Add(lName);
                }
                dCombo.Items.Insert(0, "Select a level/layer name...");
                dCombo.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Warning");
            }
        }

        private void frmFormatConsistencyGeometryonly_Load(object sender, EventArgs e)
        {
            loadLevelNames(this.cboLevelname);
            loadFeatureType(this.cboFeatureType);
            loadSearchInfo();
        }

        private void loadFeatureType(ComboBox cbo)
        {
            cbo.DataSource= Enum.GetValues(typeof(FeatureType));
            cbo.SelectedIndex = 0;
        }

        private void loadLevelNames(ComboBox cbo)
        {
            System.Collections.ArrayList niveis = getDGNLevels();
            if (niveis != null)
            {
                loadLevelsinComboBox(niveis, cbo);
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            try {
                this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
                this.Hide();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                string layername = this.cboLevelname.Text;
                FeatureType ftype = (FeatureType)this.cboFeatureType.SelectedIndex;
                SearchParametersFormatConsistency sp = setSearchP(layername, ftype);
                if (sp != null)
                {
                    this._qr.searchParams = (SearchParameters)sp;
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
```

```
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
        }

        private SearchParametersFormatConsistency setSearchP(string layername, FeatureType ftype)
        {
            SearchParametersFormatConsistency sp = null;
            try
            {
                sp = new SearchParametersFormatConsistency();
                sp.Featuretype = ftype;
                sp.LayerName = layername;
                return sp;
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); return null; }
            finally { sp = null; }
        }
    }
}
====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19115;
using ISOSpatialQuality.Classes.ISO._19109;
using ISOSpatialQuality.Classes.ISO._19103;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmLineage : Form
    {
        private LI_Lineage _lineage = null;

        private HistoryCollection _hc = null;

        public HistoryCollection historyCollection { get { return _hc; } set { _hc = value; } }
        public LI_Lineage Lineage { get { return _lineage; } set { _lineage = value; } }

        private void loadInfo()
        {
            try {
                if (_lineage != null)
                {
                    string PS = _lineage.ProjectionSystem;
                    string DT = _lineage.Datum;
                    string ES = _lineage.Ellipsoid;
                    string SC = _lineage.Scale;

                    this.txtProjectionSystem.Text = PS;
                    this.txtDatum.Text = DT;
                    this.txtEllipsoid.Text = ES;
                    this.txtScale.Text = SC;

                    if (_lineage.History != null)
                    {
                        _hc = _lineage.History;
                        UpdateDataGrid(this.dgvHistory);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally { }
        }
```

```
        public frmLineage()
        {
            InitializeComponent();
            buildColumns(this.dgvHistory);
        }

        private void cmdAddHistory_Click(object sender, EventArgs e)
        {
            frmDataHistory dh = null;
            try {
                dh = new frmDataHistory();
                if (dh.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    if (_hc == null)
                    {
                        _hc = new HistoryCollection();
                    }
                    LI_History lih = dh.History;
                    _hc.addHistory(lih);
                    this.UpdateDataGrid(this.dgvHistory);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                dh.Close();
                dh.Dispose();
            }
        }

        private DataGridViewTextBoxColumn buildTextcolumn(string colName, string HeaderText, bool IsVis
ible, string dataPropertyField)
        {
            DataGridViewTextBoxColumn retorno = null;
            try
            {
                retorno = new DataGridViewTextBoxColumn();
                retorno.Name = colName;
                retorno.HeaderText = HeaderText;
                retorno.Visible = IsVisible;
                retorno.DataPropertyName = dataPropertyField;
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private DataGridViewLinkColumn buildLinkColumn(string colName, bool isVisible, bool UseColumTex
tForLValue,
            int width, bool readOnly, string headerText, string linkText,
            string toolTip)
        {
            DataGridViewLinkColumn retorno = null;
            try {
                retorno = new DataGridViewLinkColumn();
                retorno.Name = colName;
                retorno.UseColumnTextForLinkValue = UseColumTextForLValue;
                retorno.Text = linkText;
                retorno.ToolTipText = toolTip;
                retorno.Width = width;
                retorno.ReadOnly = readOnly;
                retorno.HeaderText = headerText;

                return retorno;
            }
```

```
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void buildColumns(DataGridView dgv)
        {
            try {
                dgv.MultiSelect = false;
                dgv.AutoGenerateColumns = false;

                DataGridViewLinkColumn dgvEditRow = buildLinkColumn("EDITROW", true, true, 45, true, st
ring.Empty, "edit", "modify history information");
                DataGridViewLinkColumn dgvRemoveRow = buildLinkColumn("DELETEROW", true, true, 45, true
, string.Empty, "delete", "remove history information");


                DataGridViewTextBoxColumn dgvDate = buildTextcolumn("HistoryDate", "History Date", true
, "HISTORY_DATE");
                DataGridViewTextBoxColumn dgvHistory = buildTextcolumn("History", "History", true, "HIS
TORY");
                dgvHistory.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

                dgv.Columns.Add(dgvEditRow);
                dgv.Columns.Add(dgvRemoveRow);
                dgv.Columns.Add(dgvDate);
                dgv.Columns.Add(dgvHistory);
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void UpdateDataGrid(DataGridView dgv)
        {
            DataTable data = null;
            try {
                data = this._hc.GetDatatable();
                dgv.DataSource = data;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                data = null;
            }
        }

        private LI_Lineage buildLineage(string PS, string DT, string EL, string SC, HistoryCollection H
C)
        {
            LI_Lineage retorno = null;
            try {
                retorno = new LI_Lineage();
                retorno.ProjectionSystem = PS;
                retorno.Datum = DT;
                retorno.Ellipsoid = EL;
                retorno.Scale = SC;
                retorno.History = HC;

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }
```

```
        private void cmdOk_Click(object sender, EventArgs e)
        {
            string ProjectionSystem = string.Empty;
            string Datum = string.Empty;
            string Ellipsoid = string.Empty;
            string Scale = string.Empty;
            LI_Lineage lin = null;
            try {
                ProjectionSystem = this.txtProjectionSystem.Text;
                Datum = this.txtDatum.Text;
                Ellipsoid = this.txtEllipsoid.Text;
                Scale = this.txtScale.Text;
                lin = buildLineage(ProjectionSystem, Datum, Ellipsoid, Scale, this._hc);
                if (lin != null)
                {
                    this._lineage = lin;
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lin = null;
            }
        }

        private void frmLineage_Load(object sender, EventArgs e)
        {
            //_hc = new HistoryCollection();
            loadInfo();
        }

        private void dgvHistory_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            DataGridView dgv = null;
            try {
                dgv = (DataGridView)sender;
                if (dgv != null)
                {
                    if (dgv.Columns[e.ColumnIndex].Name == "DELETEROW")
                    {
                        this._hc.RemoveAt(e.RowIndex);
                        this.UpdateDataGrid(dgv);
                    }
                    if (dgv.Columns[e.ColumnIndex].Name == "EDITROW")
                    {
                        EditHistory(e.RowIndex);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                dgv = null;
            }
        }

        private void EditHistory(int index)
        {
            LI_History lih = null;
            try {
                lih = this._hc[index];
                if (lih != null)
                {
                    frmDataHistory fdh = new frmDataHistory();
                    fdh.History = lih;
                    if (fdh.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                    {
                        lih = fdh.History;
                        if (lih != null)
```

```
                        {
                            this._hc.InsertAt(lih, index);
                            this.UpdateDataGrid(this.dgvHistory);
                        }
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                lih = null;
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }
    }
}
==================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmQualityExpected : Form
    {
        private DBUtils _dbUtil = null;
        private ExpectedQuality _expectedQuality = null;
        private ProviderQResult _providerQuality = null;

        public ProviderQResult ProviderQuality
        {
            get { return _providerQuality; }
            set { _providerQuality = value; }
        }

        private void setComboBoxByText(ComboBox combo, string texto)
        {
            try {
                int index = combo.FindStringExact(texto);
                combo.SelectedIndex = index;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void setComboBoxIndex(ComboBox combo, int Index)
        {
            try {
                combo.SelectedIndex = Index;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void setComboBoxValue(ComboBox combo, int Value)
        {
```

```
            try {
                combo.SelectedValue = (object)Value;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        public void loadExpectedQuality()
        {
            try {
                if (_expectedQuality != null)
                {
                    setComboBoxIndex(this.cboPassFail, _expectedQuality.AcceptOption.GetHashCode());
                    setComboBoxValue(this.cboDQ_MEASUREDESC, _expectedQuality.MeasureType);
                    setComboBoxIndex(this.cboOperator, _expectedQuality.COperator.GetHashCode());
                    this.txtDQ_Value.Text = _expectedQuality.Value;
                    this.txtDQ_ValueUnit.Text = _expectedQuality.ValueUnit;
                }
                else {
                    if (_providerQuality != null)
                    {
                        this.cboPassFail.SelectedIndex = 0;
                        this.cboOperator.SelectedIndex = 0;
                        //setComboBoxByText(this.cboDQ_MEASUREDESC, _providerQuality.MeasureType);
                        int measureIndex = 0;
                        bool result = int.TryParse(_providerQuality.MeasureType, out measureIndex);
                        setComboBoxIndex(this.cboDQ_MEASUREDESC, measureIndex);

                        this.txtDQ_Value.Text = _providerQuality.Value;
                        this.txtDQ_ValueUnit.Text = _providerQuality.ValueUnit;
                    }
                    else {
                        this.cboPassFail.SelectedIndex = 0;
                        this.cboOperator.SelectedIndex = 0;
                        this.cboDQ_MEASUREDESC.SelectedIndex = 0;
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        public ExpectedQuality expectedQuality
        {
            get { return _expectedQuality; }
            set { _expectedQuality = value; }
        }

        public frmQualityExpected()
        {
            InitializeComponent();
        }

        private void frmQualityExpected_Load(object sender, EventArgs e)
        {
            _dbUtil = new DBUtils();
            loadMeasureType(this.cboDQ_MEASUREDESC);
            loadExpectedQuality();
        }

        private DataTable getMeasureType()
        {
            DataTable retorno = null;
            try
            {
                string sql = "select ID_MEASURE, MEASURE_DESCRIPTION FROM DQ_MEASURES";
                retorno = _dbUtil.getItemsFromDatabase(sql);
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
```

```
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private void loadMeasureType(ComboBox dcombo)
        {
            try
            {
                DataTable EvalTypes = getMeasureType();
                if (EvalTypes != null)
                {
                    DataRow dRow = EvalTypes.NewRow();
                    dRow[0] = 0;
                    dRow[1] = "Select a Measurement Type...";

                    if (EvalTypes.Rows.Count > 0)
                    {
                        EvalTypes.Rows.InsertAt(dRow, 0);
                    }
                    else
                    {
                        EvalTypes.Rows.Add(dRow);
                    }
                    dcombo.DisplayMember = "MEASURE_DESCRIPTION";
                    dcombo.ValueMember = "ID_MEASURE";
                    dcombo.DataSource = EvalTypes;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private int getComboIndex(ComboBox combo)
        {
            int retorno = -1;
            try {
                retorno = combo.SelectedIndex;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            ExpectedQuality dado = null;
            try {
                int acceptReject = -1;
                int measureType = -1;
                int compareOperator = -1;
                string value = string.Empty;
                string valueUnit = string.Empty;

                acceptReject = getComboIndex(this.cboPassFail);
                compareOperator = getComboIndex(this.cboOperator);
                measureType = getComboIndex(this.cboDQ_MEASUREDESC);
                //measureType = getComboSelectedValue(this.cboDQ_MEASUREDESC);
                value = this.txtDQ_Value.Text;
                valueUnit = this.txtDQ_ValueUnit.Text;

                dado = BuildExpectedQuality(acceptReject, measureType, compareOperator, value, valueUni
t);

                if (dado != null)
                {
                    _expectedQuality = dado;
```

```
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                dado = null;
            }
        }

        private int getComboSelectedValue(ComboBox combo)
        {
            int retorno = -1;
            try {
                if (combo.SelectedIndex > 0)
                {
                    bool result = int.TryParse(combo.SelectedValue.ToString(), out retorno);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private ExpectedQuality BuildExpectedQuality(int AcceptReject, int MeasureType, int compareOper
ator, string value, string valueUnit)
        {
            ExpectedQuality retorno = null;
            try {
                retorno = new ExpectedQuality();
                retorno.AcceptOption = (AcceptReject)AcceptReject;
                retorno.MeasureType = MeasureType;
                retorno.COperator = (CompareOperator)compareOperator;
                retorno.Value = value;
                retorno.ValueUnit = valueUnit;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
```

```
    public partial class frmQualityProviderResult : Form
    {
        private DBUtils _dbUtil = null;
        private ProviderQResult _providerResults = null;
        //private bool _editingInfo = false;

        public ProviderQResult ProviderResults
        {
            get { return _providerResults; }
            set { _providerResults = value; }
        }

        public frmQualityProviderResult()
        {
            InitializeComponent();
        }

        private void frmQualityProviderResult_Load(object sender, EventArgs e)
        {
            _dbUtil = new DBUtils();
            //_providerResults = new ProviderQResult();
            loadMeasureType(this.cboDQ_MEASUREDESC);
            LoadValueType(this.cboDQ_VALUETYPE);
            if (_providerResults != null)
                loadProviderResults();
        }

        private int getComboIndexByText(ComboBox combo, string text)
        {
            int retorno = -1;
            try {
                retorno = combo.FindStringExact(text);
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void loadProviderResults()
        {
            try {
                this.txtDQ_Value.Text = _providerResults.Value;
                this.txtDQ_ValueUnit.Text = _providerResults.ValueUnit;
                this.cboDQ_MEASUREDESC.SelectedIndex =   int.Parse(_providerResults.MeasureType);
                this.cboDQ_VALUETYPE.Text = _providerResults.ValueType;
                this.dtDQ_DATE.Text = _providerResults.DQ_Date;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadMeasureType(ComboBox dcombo)
        {
            try
            {
                DataTable EvalTypes = getMeasureType();
                if (EvalTypes != null)
                {
                    DataRow dRow = EvalTypes.NewRow();
                    dRow[0] = 0;
                    dRow[1] = "Select a Measurement Type...";

                    if (EvalTypes.Rows.Count > 0)
                    {
                        EvalTypes.Rows.InsertAt(dRow, 0);
                    }
                    else
                    {
                        EvalTypes.Rows.Add(dRow);
                    }
                    dcombo.DisplayMember = "MEASURE_DESCRIPTION";
```

```
                dcombo.ValueMember = "ID_MEASURE";
                dcombo.DataSource = EvalTypes;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private DataTable getMeasureType()
    {
        DataTable retorno = null;
        try
        {
            string sql = "select ID_MEASURE, MEASURE_DESCRIPTION FROM DQ_MEASURES";
            retorno = _dbUtil.getItemsFromDatabase(sql);
            return retorno;
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally
        {
            retorno = null;
        }
    }

    private void LoadValueType(ComboBox dCombo)
    {
        try
        {
            DataTable EvalTypes = getValueType();
            if (EvalTypes != null)
            {
                DataRow dRow = EvalTypes.NewRow();
                dRow[0] = 0;
                dRow[1] = "Select a Value Type...";

                if (EvalTypes.Rows.Count > 0)
                {
                    EvalTypes.Rows.InsertAt(dRow, 0);
                }
                else
                {
                    EvalTypes.Rows.Add(dRow);
                }
                dCombo.DisplayMember = "Description";
                dCombo.ValueMember = "ID";
                dCombo.DataSource = EvalTypes;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private DataTable getValueType()
    {
        DataTable retorno = null;
        try
        {
            string sql = "select ID, Description from DQ_ValueTypeEnumDomain";
            if (_dbUtil != null)
            {
                retorno = _dbUtil.getItemsFromDatabase(sql);
            }
            return retorno;
        }
        catch (Exception ex)
        {
```

```
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private int getComboSelectedIndex(ComboBox combo)
        {
            int retorno = -1;
            try {
                if (combo.SelectedIndex > -1)
                {
                    retorno = combo.SelectedIndex;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                //string measureType = getComboSelectedText(this.cboDQ_MEASUREDESC);
                string measureType = getComboSelectedIndex(this.cboDQ_MEASUREDESC).ToString();
                string valueType = getComboSelectedText(this.cboDQ_VALUETYPE);
                string valueUnit = this.txtDQ_ValueUnit.Text;
                string value = this.txtDQ_Value.Text;
                string date = this.dtDQ_DATE.Text;

                setProviderResults(measureType, valueType, valueUnit, value, date);

                if (this._providerResults != null)
                {
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private string getComboSelectedText(ComboBox dcombo)
        {
            string retorno = string.Empty;
            try {
                if (dcombo.SelectedIndex > -1)
                {
                    retorno = dcombo.Text;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return string.Empty;
            }
        }

        private void setProviderResults(string measureType, string valueType, string valueUnit, string
value, string date)
        {
            try {
```

```
                //if (!this._editingInfo)
                //{
                //    _providerResults = new ProviderQResult();
                //}
                if (_providerResults == null)
                    _providerResults = new ProviderQResult();

                _providerResults.MeasureType = measureType;
                _providerResults.ValueType = valueType;
                _providerResults.ValueUnit = valueUnit;
                _providerResults.Value = value;
                _providerResults.DQ_Date = date;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void frmQualityProviderResult_ResizeEnd(object sender, EventArgs e)
        {
            this.Width=345;
            this.Height=256;
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;

namespace ISOSpatialQuality.Forms
{
    public partial class frmQualityReport : Form
    {
        //private ScopeDescription_Feature ScFeature = null;
        private scopeDescription _scopeDesc = null;
        private ProviderQResult _providerResults = null;
        private QualityReport _qr = null;
        private DBUtils _dbUtil = null;
        private bool _editingInfo = false;

        public bool editingInfo
        {
            get {
                return _editingInfo;
            }
            set {
                _editingInfo = value;
            }
        }

        public QualityReport QualityReportInfo
        {
            get { return _qr; }
            set {
                _qr = value;
                _scopeDesc = _qr.ScopeDescription;
                _providerResults = _qr.ProviderResult;
            }
        }

        public frmQualityReport()
        {
            InitializeComponent();
```

```
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
        }

        private ScopeDescription_Feature loadScopeDescription()
        {
            ScopeDescription_Feature retorno = null;
            frmQualityScope featureQs = null;
            try {
                featureQs = new frmQualityScope();
                if (_qr != null)
                {
                    if (_qr.ScopeDescription != null)
                    {
                        featureQs.Editing = true;
                        featureQs.ScFeature = (ScopeDescription_Feature)_qr.ScopeDescription;
                    }
                    else
                    {
                        featureQs.Editing = false;
                        featureQs.ScFeature = new ScopeDescription_Feature();
                    }
                    if (featureQs.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                    {
                        retorno = featureQs.ScFeature;
                    }
                }
                return retorno;
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); return null; }
            finally {
                featureQs.Close();
                featureQs = null;
                retorno = null;
            }
        }

        private ScopeDescription_Feature loadFeatureScopeDescription()
        {
            ScopeDescription_Feature retorno = null;
            frmQualityScope featureQS = null;
            try {
                retorno = (ScopeDescription_Feature)_qr.ScopeDescription;
                featureQS = new frmQualityScope();
                featureQS.ScFeature = retorno;

                featureQS.Editing = true;
                if (_editingInfo)
                {
                    _scopeDesc = _qr.ScopeDescription;
                    featureQS.ScFeature = (ScopeDescription_Feature)_scopeDesc;
                }
                if (featureQS.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    retorno = featureQS.ScFeature;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                featureQS.Close();
                featureQS = null;
                retorno = null;
            }
        }

        private ScopeDescription_Feature setScopeDescriptionFeature()
        {
```

```
            ScopeDescription_Feature retorno = null;
            frmQualityScope featureQs = null;
            try {
                if (this._qr != null)
                {
                    featureQs = new frmQualityScope();
                    featureQs.Editing = true;
                    ScopeDescription_Feature aux = null;
                    if (this._qr.ScopeDescription != null)
                    {
                        aux = (ScopeDescription_Feature)_qr.ScopeDescription;
                        featureQs.ScFeature = aux;
                    }
                    else
                    {
                        aux = new ScopeDescription_Feature();
                        featureQs.ScFeature = aux;
                    }

                    if (featureQs.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                    {
                        retorno = featureQs.ScFeature;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                featureQs.Close();
                featureQs = null;
                retorno = null;
            }
        }

        private ScopeDescription_Dataset setScopeDescriptionDataset(string level)
        {
            ScopeDescription_Dataset retorno = null;
            try {
                retorno = new ScopeDescription_Dataset();
                retorno.level = level;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private ScopeDescription_Default setScopeDescriptionStandard(string level)
        {
            ScopeDescription_Default retorno = null;
            try
            {
                retorno = new ScopeDescription_Default();
                retorno.level = level;
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }
```

```
        private scopeDescription setScopeDescription(string level)
        {
            scopeDescription retorno = null;
            try {
                switch (level.ToUpper())
                {
                    case "FEATURE":
                        retorno = setScopeDescriptionFeature();
                        if (retorno != null)
                            retorno.level = level;
                        break;
                    case "DATASET":
                        retorno = setScopeDescriptionDataset(level);
                        if (retorno != null)
                            retorno.level = level;
                        break;
                    default:
                        retorno = setScopeDescriptionStandard(level);
                        if (retorno != null)
                            retorno.level = level;
                        break;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool isScopeDifferent(scopeDescription sd)
        {
            bool retorno = false;
            try {
                if (this._qr != null)
                {
                    if (this._qr.ScopeDescription != null)
                    {
                        if (sd.level != this._qr.ScopeDescription.level)
                        {
                            retorno = true;
                        }
                    }
                    else
                    {
                        retorno = true;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private void cmdDQScope_Click(object sender, EventArgs e)
        {
            scopeDescription sc = null;
            try {
                string scopeLevel = this.cboMD_SCOPECODE.Text;
                if (!string.IsNullOrEmpty(scopeLevel))
                {
                    sc = setScopeDescription(scopeLevel);
                }
                if (sc != null)
                {
                    _qr.ScopeDescription = sc;
                }
            }
            catch (Exception ex) {
```

```
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private int getComboboxIndexByText(ComboBox combo, string text)
        {
            int retorno = -1;
            try {
                if (string.IsNullOrEmpty(text))
                {
                    retorno = 0;
                    return retorno;
                }

                retorno = combo.FindStringExact(text);

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void frmQualityReport_Load(object sender, EventArgs e)
        {
            _scopeDesc = new ScopeDescription_Feature();
            _dbUtil = new DBUtils();
            loadQualityElement(this.cboDQ_ELEMENT);
            loadMethodType(this.cboDQ_EVALMETHODTYPE);
            this.cboInfoSource.DataSource = Enum.GetValues(typeof(InfoSource));
            this.cboInfoSource.SelectedIndex = 0;
            loadMDScopeCode(this.cboMD_SCOPECODE);
            if (_editingInfo)
            {
                loadQualityReport();
            }
            else
            {
                this._qr = new QualityReport();
            }
        }

        private void loadQualityReport()
        {
            scopeDescription sd = null;
            try {
                sd = _qr.ScopeDescription;
                if (sd != null)
                {
                    this.cboMD_SCOPECODE.SelectedIndex = getComboboxIndexByText(this.cboMD_SCOPECODE, s
d.level);
                }
                this.txtNameOfMeasure.Text = _qr.nameOfMeasure;
                this.cboDQ_ELEMENT.Text = _qr.DQ_Element;
                this.cboDQ_SUBELEMENT.Text = _qr.DQ_Subelement;
                this.txtDQ_MeasureDesc.Text = _qr.DQ_MeasureDesc;
                if (string.IsNullOrEmpty(_qr.DQ_EvalMethodType.GetHashCode().ToString()))
                {
                    this.cboDQ_EVALMETHODTYPE.SelectedIndex = -1;
                }
                else
                {
                    this.cboDQ_EVALMETHODTYPE.SelectedIndex = _qr.DQ_EvalMethodType.GetHashCode();
                }
                this.txtDQ_EvalMethodDesc.Text = _qr.DQ_EvalMethodDesc;
                this.cboInfoSource.SelectedIndex = _qr.InfoSource.GetHashCode();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                sd = null;
            }
```

```
        }

        private void loadMDScopeCode(ComboBox combo)
        {
            DataTable table = null;
            try {
                string sql = "select scopecd, displayname from md_scopecode";
                table = getMDScopeCode(sql);
                if (table != null)
                {
                    DataRow drow = table.NewRow();
                    drow["scopecd"] = "0";
                    drow["displayname"] = "Select a scope level...";
                    if (table.Rows.Count > 0)
                    {
                        table.Rows.InsertAt(drow, 0);
                    }
                    else
                    {
                        table.Rows.Add(drow);
                    }

                    combo.DataSource = table;
                    combo.DisplayMember = "DISPLAYNAME";
                    combo.ValueMember = "SCOPECD";
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                table = null;
            }
        }

        private DataTable getMDScopeCode(string sql)
        {
            DataTable retorno = null;
            try {
                if (_dbUtil != null)
                {
                    retorno = _dbUtil.getItemsFromDatabase(sql);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void loadMethodType(ComboBox dcombo)
        {
            DataTable evalTypes = null;
            try
            {
                evalTypes = getEvalMethodType();
                if (evalTypes != null)
                {
                    DataRow drow = evalTypes.NewRow();
                    drow[1] = "000";
                    drow[0] = "Select an evaluation method type...";
                    if (evalTypes.Rows.Count > 0)
                    {
                        evalTypes.Rows.InsertAt(drow, 0);
                    }
                    else
                    {
                        evalTypes.Rows.Add(drow);
                    }
                }
```

```
                dcombo.DisplayMember = "DQ_EVALUATIONMETHODTYPECODE";
                dcombo.ValueMember = "EvalMethTypeCd";
                dcombo.DataSource = evalTypes;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally
            {
                evalTypes = null;
            }
        }

        private DataTable getEvalMethodType()
        {
            DataTable retorno = null;
            try
            {
                string sql = "select DQ_EvaluationMethodTypeCode, EvalMethTypeCd, DQ_DEFINITION from DQ
_EVALMETHOD";
                retorno = _dbUtil.getItemsFromDatabase(sql);
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private void loadQualityElement(ComboBox dCombo)
        {
            DataTable dTable = null;
            try
            {
                string sql = "SELECT ID, QUALITY_DESCRIPTOR, QUALITY_INFORMATION FROM QUALITY_ELEMENTS"
;
                dTable = _dbUtil.getItemsFromDatabase(sql);
                if (dTable != null)
                {
                    dCombo.ValueMember = "ID";
                    dCombo.DisplayMember = "QUALITY_DESCRIPTOR";
                    DataRow drow = dTable.NewRow();
                    drow["ID"] = 0;
                    drow["QUALITY_DESCRIPTOR"] = "Select a quality element...";
                    dTable.Rows.InsertAt(drow, 0);
                    dCombo.DataSource = dTable;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally
            {
                dTable = null;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            frmQualityProviderResult providerResult = null;
            try {
                providerResult = new frmQualityProviderResult();
                providerResult.ProviderResults = _qr.ProviderResult;
                if (providerResult.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    _providerResults = providerResult.ProviderResults;
                    _qr.ProviderResult = providerResult.ProviderResults;
```

```
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                providerResult.Close();
                providerResult = null;
            }
        }

        private void cboDQ_ELEMENT_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox dcombo = null;
            DataTable dtable = null;
            try
            {
                dcombo = (ComboBox)sender;
                if (dcombo != null)
                {
                    if (dcombo.SelectedIndex > 0)
                    {
                        string ID = dcombo.SelectedValue.ToString();
                        dtable = getQualitySubElements(ID);
                        if (dtable != null)
                        {
                            this.cboDQ_SUBELEMENT.DataSource = dtable;
                            this.cboDQ_SUBELEMENT.DisplayMember = "SUBELEMENTS_DESCRIPTOR";
                            this.cboDQ_SUBELEMENT.ValueMember = "ID";
                            this.cboDQ_SUBELEMENT.SelectedIndex = 0;
                        }
                    }
                    else
                    {
                        this.cboDQ_SUBELEMENT.DataSource = null;
                        this.cboDQ_SUBELEMENT.ValueMember = string.Empty;
                        this.cboDQ_SUBELEMENT.DisplayMember = string.Empty;
                        this.cboDQ_SUBELEMENT.Items.Clear();
                        this.cboDQ_SUBELEMENT.Items.Add("Select a quality subelement...");
                        this.cboDQ_SUBELEMENT.SelectedIndex = 0;
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally
            {
                dcombo = null;
                dtable = null;
            }
        }

        private DataTable getQualitySubElements(string ID)
        {
            DataTable dTable = null;
            try
            {
                string sql = "SELECT ID, SUBELEMENTS_DESCRIPTOR, SUBELEMENTS_INFORMATION FROM QUALITY_S
UBELEMENTS WHERE ID_QUALITY_ELEMENT = " + ID;
                if (_dbUtil != null)
                {
                    dTable = _dbUtil.getItemsFromDatabase(sql);
                    if (dTable != null)
                    {
                        DataRow drow = dTable.NewRow();
                        drow[0] = 0;
                        drow[1] = "Select a quality subelement...";
                        dTable.Rows.InsertAt(drow, 0);
                    }
                }
                return dTable;
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                dTable = null;
            }
        }

        private bool setQualityReportElements(string nameOfMeasure,string qualityElement, string qualit
ySubElement, string measureDescription, int evaluationType, string evaluationDescription, int informati
onSource)
        {
            bool retorno = false;
            try {

                if (this._qr != null)
                {
                    _qr.nameOfMeasure = nameOfMeasure;
                    _qr.DQ_Element = qualityElement;
                    _qr.DQ_Subelement = qualitySubElement;
                    _qr.DQ_MeasureDesc = measureDescription;
                    _qr.DQ_EvalMethodType = (DQ_EvaluationMethodTypecode)evaluationType;
                    _qr.DQ_EvalMethodDesc = evaluationDescription;
                    _qr.InfoSource = (InfoSource)informationSource;
                }

                retorno = true;

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private string getComboSelectedText(ComboBox combo)
        {
            string retorno = string.Empty;
            try {
                retorno = combo.Text;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return string.Empty;
            }
        }

        private int getComboSelectedIndex(ComboBox combo)
        {
            int retorno = -1;
            try {
                retorno = combo.SelectedIndex;
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private int getComboSelectedValue(ComboBox combo)
        {
            int retorno = -1;
            try {
                if (combo.SelectedIndex > -1)
                {
                    if (!string.IsNullOrEmpty(combo.SelectedValue.ToString()))
                    {
                        retorno = int.Parse(combo.SelectedValue.ToString());
```

```
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                string nameOfMeasure = this.txtNameOfMeasure.Text;
                string dq_element = getComboSelectedText(this.cboDQ_ELEMENT);
                string dq_subelement = getComboSelectedText(this.cboDQ_SUBELEMENT);
                string dq_measuredesc = this.txtDQ_MeasureDesc.Text;
                int dq_evaluationType = getComboSelectedIndex(this.cboDQ_EVALMETHODTYPE);
                string dq_evalDescription = this.txtDQ_EvalMethodDesc.Text;
                int infoSource = getComboSelectedIndex(this.cboInfoSource);
                if (setQualityReportElements(nameOfMeasure,dq_element, dq_subelement, dq_measuredesc, d
q_evaluationType, dq_evalDescription, infoSource))
                {
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Could not create Quality Report Info.", "Error");
                    this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
                    this.Close();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cboInfoSource_SelectedIndexChanged(object sender, EventArgs e)
        {
            ComboBox combo = null;
            try {
                combo = (ComboBox)sender;
                if (combo != null)
                {
                    if (combo.SelectedIndex == 1)
                    {
                        this.cmdEval.Enabled = true;
                    }
                    else
                    {
                        this.cmdEval.Enabled = false;
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                combo = null;
            }
        }

        private void frmQualityReport_Resize(object sender, EventArgs e)
        {
            this.Width=431;
            this.Height=473;
        }

        private void set_QuantitativeAttributeParameters()
        {
            frmQualityExpected qe = null;
            try {
                qe = new frmQualityExpected();
```

```
                qe.ProviderQuality = this._qr.ProviderResult;
                qe.expectedQuality = (ExpectedQuality)this._qr.expectedQuality;
                if (qe.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    _qr.expectedQuality = qe.expectedQuality;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                qe.Close();
                qe = null;
            }
        }

        private void set_ExpectedFormat()
        {
            frmUserExpectedFormat expectedF = null;
            try {
                expectedF = new frmUserExpectedFormat();
                expectedF.expectedFormat = (ExpectedFormatConsistencyCAD)this._qr.expectedQuality;
                if (expectedF.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    _qr.expectedQuality = expectedF.expectedFormat;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                expectedF.Close();
                expectedF = null;
            }
        }

        private void set_CompletenessRecordExpectation()
        {
            //frmQualityExpected fqe = null;
            frmUserExpectedCompletenessRecord fuecr = null;
            try {
                fuecr = new frmUserExpectedCompletenessRecord();
                fuecr.qualityReport = this._qr;
                if (fuecr.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    this._qr.expectedQuality = fuecr.qualityReport.expectedQuality;
                }

            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                fuecr.Close();
                fuecr = null;
                //fqe.Close();
                //fqe = null;
            }
        }

        private void set_CompletenessSchemaExpectation()
        {
            frmCompletenessSchemaAcceptanceCriteriaForm csacf = null;
            try {
                csacf = new frmCompletenessSchemaAcceptanceCriteriaForm();
                csacf.qualityReport = this._qr;
                if (csacf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    this._qr.expectedQuality = csacf.qualityReport.expectedQuality;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
```

```
            }
            finally {
                csacf.Close();
                csacf = null;
            }
        }

        private void set_CompletenessExpectation()
        {
            CompletenessAssessment ca = CompletenessAssessment.undefined;
            frmCompletenessChoice cc = null;
            try {
                cc = new frmCompletenessChoice();
                cc.qualityReport = this._qr;
                if (cc.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    ca = cc.completenessAssessment;
                    this._qr.expectedQuality = cc.qualityReport.expectedQuality;
                }
                switch (ca)
                {
                    case CompletenessAssessment.undefined:
                        break;
                    case CompletenessAssessment.CompletenessSchema:
                        set_CompletenessSchemaExpectation();
                        break;
                    case CompletenessAssessment.CompletenessRecord:
                        set_CompletenessRecordExpectation();
                        break;
                    default:
                        break;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                cc = null;
            }
        }

        private void cmdExpectedQuality_Click(object sender, EventArgs e)
        {
            try {
                string nameOfMeasure = this.txtNameOfMeasure.Text;
                string dq_element = getComboSelectedText(this.cboDQ_ELEMENT);
                string dq_subelement = getComboSelectedText(this.cboDQ_SUBELEMENT);
                string dq_measuredesc = this.txtDQ_MeasureDesc.Text;
                int dq_evaluationType = getComboSelectedIndex(this.cboDQ_EVALMETHODTYPE);
                string dq_evalDescription = this.txtDQ_EvalMethodDesc.Text;
                int infoSource = getComboSelectedIndex(this.cboInfoSource);
                if (setQualityReportElements(nameOfMeasure, dq_element, dq_subelement, dq_measuredesc,
dq_evaluationType, dq_evalDescription, infoSource))
                {
                    switch (this._qr.subElement)
                    {
                        case QualitySubelement.DQ_AbsoluteExternalPositionalAccuracy:
                            break;
                        case QualitySubelement.DQ_AccuracyOfATimeMeasurement:
                            break;
                        case QualitySubelement.DQ_CompletenessComission:
                        case QualitySubelement.DQ_CompletenessOmission:
                            set_CompletenessExpectation();
                            break;
                        case QualitySubelement.DQ_ConceptualConsistency:
                            set_QuantitativeAttributeParameters();
                            break;
                        case QualitySubelement.DQ_DomainConsistency:
                            set_QuantitativeAttributeParameters();
                            break;
                        case QualitySubelement.DQ_FormatConsistency:
                            set_ExpectedFormat();
                            break;
                        case QualitySubelement.DQ_GriddedDataPositionalAccuracy:
```

```
                        break;
                    case QualitySubelement.DQ_NonQuantitativeAttributeAccuracy:
                        set_QuantitativeAttributeParameters();
                        break;
                    case QualitySubelement.DQ_QuantitativeAttributeAccuracy:
                        //set_QuantitativeAttributeParameters();
                        break;
                    case QualitySubelement.DQ_RelativeInternalPositionalAccuracy:
                        break;
                    case QualitySubelement.DQ_TemporalConsistency:
                        break;
                    case QualitySubelement.DQ_TemporalValidity:
                        break;
                    case QualitySubelement.DQ_ThematicClassificationCorrectness:
                        break;
                    case QualitySubelement.DQ_TopologicalConsistency:
                        break;
                    default:
                        break;
                }
            }

        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private scopeDescription setScopeLevel(string level)
    {
        scopeDescription sd = null;
        try {
            switch (level.ToUpper())
            {
                case "FEATURE":
                    sd = new ScopeDescription_Feature();
                    break;
                case "DATASET":
                    sd = new ScopeDescription_Dataset();
                    break;
                default:
                    sd = new ScopeDescription_Default();
                    break;
            }
            if (sd != null)
            {
                sd.level = level;
            }
            return sd;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally {
            sd = null;
        }
    }

    private void cboMD_SCOPECODE_SelectedIndexChanged(object sender, EventArgs e)
    {
        ComboBox combo = null;
        scopeDescription sd = null;
        try {
            combo = (ComboBox)sender;
            if (combo.SelectedIndex > 0)
            {
                sd = setScopeLevel(combo.Text);
                if (isScopeDifferent(sd))
                    this._qr.ScopeDescription = sd;
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
```

```
            }
            finally {
                combo = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality.Classes;

namespace ISOSpatialQuality.Forms
{
    public partial class frmQualityScope : Form
    {
        private ScopeDescription_Feature _scFeature = null;
        private bool _editing = false;

        public bool Editing { get { return _editing; } set { _editing = value; } }

        public ScopeDescription_Feature ScFeature
        {
            get { return _scFeature; }
            set { _scFeature = value; }
        }

        public frmQualityScope()
        {
            InitializeComponent();
        }

        private bool setScopeDescription(string Description, string feature, string minX, string minY,
string maxX, string maxY)
        {
            try {
                _scFeature.Description = Description;
                _scFeature.feature = feature;
                _scFeature.minX = minX;
                _scFeature.minY = minY;
                _scFeature.maxX = maxX;
                _scFeature.maxY = maxY;

                return true;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                string description = this.txtDescription.Text;
                string feature = this.txtFeatureName.Text;
                string minx = this.txtMinX.Text;
                string miny = this.txtMinY.Text;
                string maxx = this.txtMaxX.Text;
                string maxy = this.txtMaxY.Text;

                if (setScopeDescription(description, feature, minx, miny, maxx, maxy))
                {
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex) {
```

```
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void frmQualityScope_Load(object sender, EventArgs e)
        {
            //_scFeature = new ScopeDescription_Feature();
            if (_editing)
            {
                if (_scFeature != null)
                    loadScope();
            }
            else
            {
                this._scFeature = new ScopeDescription_Feature();
            }
        }

        private void loadScope()
        {
            try {
                this.txtDescription.Text = _scFeature.Description;
                this.txtFeatureName.Text = _scFeature.feature;
                this.txtMinX.Text = _scFeature.minX;
                this.txtMinY.Text = _scFeature.minY;
                this.txtMaxX.Text = _scFeature.maxX;
                this.txtMaxY.Text = _scFeature.maxY;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Close();
        }

        private void cmdCancel_Click_1(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void frmQualityScope_ResizeEnd(object sender, EventArgs e)
        {
            this.Width=355;
            this.Height=295;
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Resources;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.ISO._19115;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Utilities;
using System.Xml.Linq;
using Microsoft.VisualBasic.Devices;
using BIM = Bentley.Interop.MicroStationDGN;
using BIX = Bentley.Interop.Xft;

namespace ISOSpatialQuality.Forms
```

```
{
    public partial class frmQualitySetup : Form
    {
        private QualityReportEngine _qe = null;
        private LI_Lineage _lineage = null;

        public LI_Lineage Lineage { get { return _lineage; } set { _lineage = value; } }

        public frmQualitySetup()
        {
            InitializeComponent();
            BuildColumns(this.dgvQualityStatement);
        }

        private void updateDataGrid(DataGridView dgv)
        {
            DataTable retorno = null;
            try {
                retorno = _qe.getDataTable();
                dgv.DataSource = retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                retorno = null;
            }
        }

        private void cmdNewQualityStatement_Click(object sender, EventArgs e)
        {
            frmQualityReport qReport = null;
            try {
                qReport = new frmQualityReport();
                if (qReport.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    QualityReport qr = qReport.QualityReportInfo;
                    _qe.QualityCollection.addQualityReport(qr);
                    updateDataGrid(this.dgvQualityStatement);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                qReport.Close();
                qReport = null;
            }
        }

        private DataGridViewTextBoxColumn buildTextcolumn(string colName, string HeaderText, bool IsVis
ible, string dataPropertyField)
        {
            DataGridViewTextBoxColumn retorno = null;
            try
            {
                retorno = new DataGridViewTextBoxColumn();
                retorno.Name = colName;
                retorno.HeaderText = HeaderText;
                retorno.Visible = IsVisible;
                retorno.DataPropertyName = dataPropertyField;
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }
```

```
        private void BuildColumns(DataGridView dgv)
        {
            DataGridViewCheckBoxColumn dgvProcessRow = null;
            DataGridViewButtonColumn dgvSearchCriteria = null;
            DataGridViewLinkColumn dgvEditRow = null;
            //DataGridViewImageColumn dgvReportCol = null;
            DataGridViewImageColumn dgvUserResult = null;
            try
            {
                dgv.AutoGenerateColumns = false;
                dgv.MultiSelect = false;
                dgv.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
                dgv.VirtualMode = true;

                dgvUserResult = new DataGridViewImageColumn();
                dgvUserResult.DefaultCellStyle.BackColor = Color.White;
                dgvUserResult.DefaultCellStyle.ForeColor = Color.Black;
                dgvUserResult.DefaultCellStyle.SelectionBackColor = Color.White;
                dgvUserResult.DefaultCellStyle.SelectionForeColor = Color.Black;
                dgvUserResult.Resizable = DataGridViewTriState.False;
                dgvUserResult.Width = 32;
                dgvUserResult.Visible = true;

                dgvProcessRow = new DataGridViewCheckBoxColumn();
                dgvProcessRow.Name = "PROCESSROW";
                dgvProcessRow.HeaderText = string.Empty;
                dgvProcessRow.Visible = true;
                dgvProcessRow.Width = 25;
                dgvProcessRow.ToolTipText = "Check this box if you want to process this quality element
...";
                dgvProcessRow.DataPropertyName = "PROCESS";
                dgvProcessRow.TrueValue = true.GetHashCode();
                dgvProcessRow.FalseValue = false.GetHashCode();
                dgvProcessRow.IndeterminateValue = -1;

                dgvSearchCriteria = new DataGridViewButtonColumn();
                dgvSearchCriteria.Name = "ENTERDETAIL";
                dgvSearchCriteria.HeaderText = string.Empty;
                dgvSearchCriteria.Text = "...";
                dgvSearchCriteria.ToolTipText = "Enter search criteria for this quality element";
                dgvSearchCriteria.UseColumnTextForButtonValue = true;
                dgvSearchCriteria.Visible = true;
                dgvSearchCriteria.Width = 30;

                dgvEditRow = new DataGridViewLinkColumn();
                dgvEditRow.Name = "EDITROW";
                dgvEditRow.UseColumnTextForLinkValue = true;
                dgvEditRow.Text = "edit";
                dgvEditRow.ToolTipText = "modify quality statement information";
                dgvEditRow.Visible = true;
                dgvEditRow.Width = 45;
                dgvEditRow.ReadOnly = true;
                dgvEditRow.HeaderText = string.Empty;

                DataGridViewTextBoxColumn dgvDQMeasureID = buildTextcolumn("DQ_MEASUREIDEP", "DQ_Measur
eID", false, "DQ_MEASUREID");
                DataGridViewTextBoxColumn dgvDQElementID = buildTextcolumn("ELEMENT_IDEP", "Element ID"
, false, "ELEMENT_ID");
                DataGridViewTextBoxColumn dgvDQElement = buildTextcolumn("DQ_ELEMENTEP", "DQ_Element",
true, "DQ_ELEMENT");
                DataGridViewTextBoxColumn dgvDQSubelement = buildTextcolumn("DQ_SUBELEMENTEP", "DQ_SubE
lement", true, "DQ_SUBELEMENT");
                DataGridViewTextBoxColumn dgvDQMeasureDesc = buildTextcolumn("DQ_MEASUREDESCEP", "DQ_Me
asureDesc", true, "DQ_MEASUREDESC");
                DataGridViewTextBoxColumn dgvDQScope = buildTextcolumn("DQ_SCOPEEP", "DQ_Scope", false,
 "DQ_SCOPE");
                DataGridViewTextBoxColumn dgvDQEvalMethodType = buildTextcolumn("DQ_EVALMETHODTYPEEP",
"DQ_EvalMethodType", true, "DQ_EVALMETHODTYPE");
                DataGridViewTextBoxColumn dgvDQEvalMethodDesc = buildTextcolumn("DQ_EVALMETHODDESCEP",
"DQ_EvalMethodDesc", true, "DQ_EVALMETHODDESC");
                DataGridViewTextBoxColumn dgvDQValueType = buildTextcolumn("DQ_VALUETYPEEP", "DQ_ValueT
ype", false, "DQ_VALUETYPE");
                DataGridViewTextBoxColumn dgvDQValue = buildTextcolumn("DQ_VALUEEP", "DQ_Value", false,
 "DQ_VALUE");
```

```
                DataGridViewTextBoxColumn dgvDQValueUnit = buildTextcolumn("DQ_VALUEUNITEP", "DQ_ValueU
nit", false, "DQ_VALUEUNIT");
                DataGridViewTextBoxColumn dgvLayerName = buildTextcolumn("LAYERNAMEEP", "LayerName", fa
lse, "LAYERNAME");
                DataGridViewTextBoxColumn dgvSource = buildTextcolumn("SOURCEEP", "Source", false, "SOU
RCE");
                DataGridViewTextBoxColumn dgvMinX = buildTextcolumn("MINXEP", "MinX", false, "MINX");
                DataGridViewTextBoxColumn dgvMinY = buildTextcolumn("MINYEP", "MinY", false, "MINY");
                DataGridViewTextBoxColumn dgvMaxX = buildTextcolumn("MAXXEP", "MaxX", false, "MAXX");
                DataGridViewTextBoxColumn dgvMaxY = buildTextcolumn("MAXYEP", "MaxY", false, "MAXY");
                DataGridViewTextBoxColumn dgvDQDate = buildTextcolumn("DQ_DATEEP", "DQ_Date", false, "D
Q_DATE");
                DataGridViewTextBoxColumn dgvDQConformanceLevel = buildTextcolumn("DQ_CONFORMANCELEVELE
P", "DQ_ConformanceLevel", false, "DQ_CONFORMANCELEVEL");
                DataGridViewTextBoxColumn dgvDQConformanceParameter = buildTextcolumn("DQ_CONFORMANCEPA
RAMETEREP", "DQ_ConformanceParameter", false, "DQ_CONFORMANCEPARAMETER");
                DataGridViewTextBoxColumn dgvDQScopeCD = buildTextcolumn("DQ_SCOPECDEP", "DQ_ScopeCode"
, false, "DQ_SCOPECD");

                //dgv.Columns.Add(dgvReportCol);
                dgv.Columns.Add(dgvUserResult);
                dgv.Columns.Add(dgvProcessRow);
                dgv.Columns.Add(dgvEditRow);
                dgv.Columns.Add(dgvSearchCriteria);
                dgv.Columns.Add(dgvDQElement);
                dgv.Columns.Add(dgvDQSubelement);
                dgv.Columns.Add(dgvDQMeasureDesc);
                dgv.Columns.Add(dgvDQScope);
                dgv.Columns.Add(dgvDQEvalMethodType);
                dgv.Columns.Add(dgvDQValueType);
                dgv.Columns.Add(dgvDQValue);
                dgv.Columns.Add(dgvDQValueUnit);
                dgv.Columns.Add(dgvDQConformanceLevel);

                dgv.Columns.Add(dgvDQMeasureID);
                dgv.Columns.Add(dgvLayerName);
                dgv.Columns.Add(dgvMinX);
                dgv.Columns.Add(dgvMinY);
                dgv.Columns.Add(dgvMaxX);
                dgv.Columns.Add(dgvMaxY);
                dgv.Columns.Add(dgvDQElementID);
                dgv.Columns.Add(dgvSource);
                dgv.Columns.Add(dgvDQScopeCD);

                dgv.Columns.Add(dgvDQConformanceParameter);

                dgvDQEvalMethodDesc.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;
                dgv.Columns.Add(dgvDQEvalMethodDesc);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                dgvProcessRow.Dispose();
                dgvSearchCriteria.Dispose();
                dgvEditRow.Dispose();
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void frmQualitySetup_Load(object sender, EventArgs e)
        {
            //this._qeiCol = new QualityElementInfoCollection();
            //this._qe = new QualityEngine();
            this._qe = new QualityReportEngine();
            getDatasetInformation();
        }

        private void getDatasetInformation()
```

```
        {
            try {
                string fileName = string.Empty;
                string filePath = string.Empty;
                getCurrentDesignFileName(out fileName, out filePath);

                this.lblFILENAME.Text = fileName;
                this.lblFILEPATH.Text = filePath;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void getCurrentDesignFileName(out string FileName, out string FilePath)
        {
            string retorno = string.Empty;
            FileName = string.Empty;
            FilePath = string.Empty;
            Bentley.Interop.MicroStationDGN.Application dMstn = null;
            try
            {
                dMstn = ISOSpatialQuality.ComApp;
                string fileName = dMstn.ActiveDesignFile.FullName.ToString();
                retorno = System.IO.Path.GetFileName(fileName);
                FileName = retorno;
                FilePath = System.IO.Path.GetFullPath(fileName);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                FileName = string.Empty;
                FilePath = string.Empty;
            }
            finally
            {
                dMstn = null;
            }
        }

        private void cmdSaveQualityStatement_Click(object sender, EventArgs e)
        {
            XDocument file = null;
            try {
                saveFileDLG.Filter = "XML File|*.xml";
                saveFileDLG.FilterIndex = 0;
                saveFileDLG.Title = "Save Quality Statement";
                saveFileDLG.InitialDirectory = System.IO.Path.GetDirectoryName(this.lblFILEPATH.Text);
                if (saveFileDLG.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    if (!string.IsNullOrEmpty(saveFileDLG.FileName))
                    {
                        XElement report = _qe.getXMLQualityInfo();
                        XElement source = null;
                        if (this._lineage != null)
                        {
                            source = this._lineage.GetXMLInfo();
                        }

                        XElement lineage = new XElement("lineage");

                        if (source != null)
                        {
                            lineage.Add(source);
                        }

                        file = new XDocument();

                        XElement dataQuality = new XElement("DQ_DataQuality");

                        dataQuality.Add(lineage);
                        dataQuality.Add(report);

                        //file.Add(lineage);
```

```
                        //file.Add(report);

                        file.Add(dataQuality);

                        //file = _qe.getXMLQualityInfo();
                        file.Save(saveFileDLG.FileName);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                file = null;
            }
        }

        private void cmdLoadProviderStatement_Click(object sender, EventArgs e)
        {
            try {
                openFileDialog.Title = "Data Quality File";
                openFileDialog.Filter = "XML File|*.xml";
                openFileDialog.FilterIndex = 0;
                openFileDialog.InitialDirectory = System.IO.Path.GetDirectoryName(this.lblFILEPATH.Text
);
                if (openFileDialog.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    _qe.loadDataQualityInfo(openFileDialog.FileName.ToString());
                    _lineage = new LI_Lineage(openFileDialog.FileName.ToString());
                    updateDataGrid(this.dgvQualityStatement);
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private bool editQualityReport(int QualityReportIndex)
        {
            frmQualityReport qreportForm = null;
            QualityReport oldReport = null;
            bool retorno = false;
            try {
                qreportForm = new frmQualityReport();
                oldReport = _qe.QualityCollection[QualityReportIndex];
                qreportForm.QualityReportInfo = oldReport;
                qreportForm.editingInfo = true;
                if (qreportForm.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    QualityReport newReport = qreportForm.QualityReportInfo;
                    if (newReport != null)
                    {
                        _qe.QualityCollection.replaceQualityReport(newReport, QualityReportIndex);
                        retorno = true;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                qreportForm.Close();
                oldReport = null;
                qreportForm = null;
            }
        }

        private SearchParametersQuantitativeAttribute getSearchParam(QualityReport qr)
        {
            frmQuantitativeAttribute fqa = null;
            SearchParametersQuantitativeAttribute retorno = null;
            try {
```

```
                fqa = new frmQuantitativeAttribute();
                fqa.QualityReport = qr;
                retorno = (SearchParametersQuantitativeAttribute)qr.searchParams;
                if (fqa.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    retorno = (SearchParametersQuantitativeAttribute)fqa.QualityReport.searchParams;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                fqa.Close();
                fqa = null;
                retorno = null;
            }
        }

        private bool getSearchParam_QuantitativeAttributes(QualityReport qr, out SearchParametersQuanti
tativeAttribute spqa)
        {
            bool retorno = false;
            frmQuantitativeAttribute fqa = null;
            spqa = (SearchParametersQuantitativeAttribute)qr.searchParams;
            try {
                fqa = new frmQuantitativeAttribute();
                fqa.QualityReport = qr;
                if (fqa.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    spqa = (SearchParametersQuantitativeAttribute)fqa.QualityReport.searchParams;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                fqa.Close();
                fqa = null;
            }
        }

        //private bool getSearchParam_CompletenessRecord(QualityReport qr, out SearchParametersComplete
nessRecord spcr)
        //{
        //    try { }
        //    catch (Exception ex) { }
        //    finally { }
        //}

        private bool getSearchParam_CompletenessSchema(QualityReport qr, out SearchParametersCompletene
ssSchema spcs)
        {
            bool retorno = false;
            frmComissionOmissionSchema fcos = null;
            spcs = (SearchParametersCompletenessSchema)qr.searchParams;
            try {

                fcos = new frmComissionOmissionSchema();
                fcos.qualityReport = qr;
                if (fcos.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    spcs = (SearchParametersCompletenessSchema)fcos.qualityReport.searchParams;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
```

```
            }
            finally {
                fcos.Close();
                fcos = null;
            }
        }

        private bool getSearchParam_CompletenessRecord(QualityReport qr, out SearchParametersCompletene
ssRecord spcr)
        {
            bool retorno = false;
            frmComissionOmissionRecord fcor = null;
            spcr = (SearchParametersCompletenessRecord)qr.searchParams;
            try {
                fcor = new frmComissionOmissionRecord();
                fcor.qualityReport = qr;
                if (fcor.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    spcr = (SearchParametersCompletenessRecord)fcor.qualityReport.searchParams;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                fcor.Close();
                fcor = null;
            }
        }

        private bool getSearchParam_Completeness(QualityReport qr, out SearchParametersCompleteness spc
)
        {
            //CompletenessAssessment ca = CompletenessAssessment.undefined;
            spc = (SearchParametersCompleteness)qr.searchParams;
            bool retorno = false;
            try {
                if (qr.expectedQuality != null)
                {
                    ExpectedCompleteness ec = (ExpectedCompleteness)qr.expectedQuality;
                    if (ec != null)
                    {
                        switch (ec.completenessAssessment)
                        {
                            case CompletenessAssessment.undefined:
                                break;
                            case CompletenessAssessment.CompletenessSchema:
                                SearchParametersCompletenessSchema spcs = (SearchParametersCompleteness
Schema)spc;
                                retorno = getSearchParam_CompletenessSchema(qr,out spcs);
                                spc = spcs;
                                break;
                            case CompletenessAssessment.CompletenessRecord:
                                SearchParametersCompletenessRecord spcr = (SearchParametersCompleteness
Record)spc;
                                retorno = getSearchParam_CompletenessRecord(qr, out spcr);
                                spc = spcr;
                                break;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private bool getSearchParam_DomainConsistency(QualityReport qr, out SearchParametersDomainConsi
stency spdc)
```

```
        {
            frmExpectedFeatureSchema fefs = null;
            bool retorno = false;
            spdc = (SearchParametersDomainConsistency)qr.searchParams;
            try {
                fefs = new frmExpectedFeatureSchema();
                fefs.qualityReport = qr;
                if (fefs.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    spdc = (SearchParametersDomainConsistency)fefs.qualityReport.searchParams;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                fefs.Close();
                fefs = null;
            }
        }

        private bool getSearchP_FormatConsistency(QualityReport qr, out SearchParametersFormatConsisten
cy spfc)
        {
            bool retorno = false;
            frmFormatConsistencyGeometryonly dform = null;
            spfc = (SearchParametersFormatConsistency)qr.searchParams;
            try {
                dform = new frmFormatConsistencyGeometryonly();
                dform.qualityReport = qr;
                if (dform.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                {
                    spfc = (SearchParametersFormatConsistency)dform.qualityReport.searchParams;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); return false; }
            finally {
                dform.Close();
                dform = null;
            }
        }

        private bool setSearchParam(QualityReport qr, out SearchParameters spqa)
        {
            bool retorno = false;
            spqa = qr.searchParams;
            try {
                switch (qr.DQ_Subelement.ToUpper())
                {
                    case "QUANTITATIVE ATTRIBUTE ACCURACY":
                        SearchParametersQuantitativeAttribute dSPQA = null;
                        //spqa = (SearchParametersQuantitativeAttribute)qr.searchParams;
                        dSPQA = (SearchParametersQuantitativeAttribute)spqa;
                        //spqa = (SearchParametersQuantitativeAttribute)qr.searchParams;
                        retorno = getSearchParam_QuantitativeAttributes(qr, out dSPQA);
                        spqa = dSPQA;
                        break;
                    case "CONCEPTUAL CONSISTENCY":
                        SearchParametersFormatConsistency spfc = null;
                        retorno = getSearchP_FormatConsistency(qr, out spfc);
                        spqa = spfc;
                        break;
                    case "FORMAT CONSISTENCY":
                        break;
                    default:
                        retorno = false;
                        break;
                }
                return retorno;
```

```
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            //finally { }
        }

        private SearchParameters setSearchParam(QualityReport qr)
        {
            SearchParameters searchP = null;
            try {
                switch (qr.DQ_Subelement.ToUpper())
                {
                    case "QUANTITATIVE ATTRIBUTE ACCURACY":
                        searchP = getSearchParam(qr);
                        break;
                    default:
                        break;
                }
                return searchP;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                searchP = null;
            }
        }

        private bool setSearchParameters(QualityReport qr, out SearchParameters sp)
        {
            //sp = null;
            sp = qr.searchParams;
            bool retorno = false;
            try {
                switch (qr.subElement)
                {
                    case QualitySubelement.DQ_AbsoluteExternalPositionalAccuracy:
                        break;
                    case QualitySubelement.DQ_AccuracyOfATimeMeasurement:
                        break;
                    case QualitySubelement.DQ_CompletenessComission:
                    case QualitySubelement.DQ_CompletenessOmission:
                        SearchParametersCompleteness spc = null;
                        spc = (SearchParametersCompleteness)sp;
                        retorno = getSearchParam_Completeness(qr, out spc);
                        sp = spc;
                        break;
                    case QualitySubelement.DQ_ConceptualConsistency:
                        SearchParametersFormatConsistency spfc = null;
                        spfc = (SearchParametersFormatConsistency)sp;
                        retorno = getSearchP_FormatConsistency(qr, out spfc);
                        sp = spfc;
                        break;
                    case QualitySubelement.DQ_DomainConsistency:
                        SearchParametersDomainConsistency spdc = null;
                        spdc = (SearchParametersDomainConsistency)sp;
                        retorno = getSearchParam_DomainConsistency(qr, out spdc);
                        sp = spdc;
                        break;
                    case QualitySubelement.DQ_FormatConsistency:
                        break;
                    case QualitySubelement.DQ_GriddedDataPositionalAccuracy:
                        break;
                    case QualitySubelement.DQ_NonQuantitativeAttributeAccuracy:
                        SearchParametersQuantitativeAttribute spqa = null;
                        spqa = (SearchParametersQuantitativeAttribute)sp;
                        retorno = getSearchParam_QuantitativeAttributes(qr, out spqa);
                        sp = spqa;
                        break;
                    case QualitySubelement.DQ_QuantitativeAttributeAccuracy:
                        //SearchParametersQuantitativeAttribute spqa = null;
```

```
                    //spqa = (SearchParametersQuantitativeAttribute)sp;
                    //retorno = getSearchParam_QuantitativeAttributes(qr, out spqa);
                    //sp = spqa;
                    break;
                case QualitySubelement.DQ_RelativeInternalPositionalAccuracy:
                    break;
                case QualitySubelement.DQ_TemporalConsistency:
                    break;
                case QualitySubelement.DQ_TemporalValidity:
                    break;
                case QualitySubelement.DQ_ThematicClassificationCorrectness:
                    break;
                case QualitySubelement.DQ_TopologicalConsistency:
                    break;
                default:
                    break;
            }
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return false;
        }
    }

    private void editSearchParam(int QualityIndex)
    {
        try {
            if (QualityIndex > -1)
            {
                QualityReport qr = this._qe.QualityCollection[QualityIndex];
                //qr.searchParams = setSearchParam(qr);
                //qr.DetectedElements = null;
                //this._qe.QualityCollection.replaceQualityReport(qr, QualityIndex);
                //updateDataGrid(this.dgvQualityStatement);
                SearchParameters sp = qr.searchParams;
                //bool retorno = setSearchParam(qr, out sp);
                bool retorno = setSearchParameters(qr, out sp);
                if (retorno)
                {
                    qr.searchParams = sp;
                    qr.DetectedElements = null;
                    this._qe.QualityCollection.replaceQualityReport(qr, QualityIndex);
                    updateDataGrid(this.dgvQualityStatement);
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void dgvQualityStatement_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        DataGridView dgv = null;
        try {
            dgv = (DataGridView)sender;
            if (dgv != null)
            {
                if (dgv.Columns[e.ColumnIndex].Name.ToString() == "EDITROW")
                {
                    if (editQualityReport(e.RowIndex))
                    {
                        updateDataGrid(this.dgvQualityStatement);
                    }
                }
                if (dgv.Columns[e.ColumnIndex].Name.ToString().ToUpper() == "ENTERDETAIL")
                {
                    editSearchParam(e.RowIndex);
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
```

```
            }
            finally {
                dgv = null;
            }
        }

        private void cmdExecEvaluation_Click(object sender, EventArgs e)
        {
            bool anyProcessed = false;
            frmShowEvalResult dREsults = null;
            try {
                this.statusText.Text = "Processing dataset... please wait.";
                Application.DoEvents();
                this.Cursor = Cursors.WaitCursor;
                foreach (DataGridViewRow dgvr in this.dgvQualityStatement.Rows)
                {
                    DataGridViewCheckBoxCell dgvcBoxCell = (DataGridViewCheckBoxCell)dgvr.Cells["PROCES
SROW"];
                    if (dgvcBoxCell != null)
                    {
                        if (dgvcBoxCell.Value == null)
                        {
                            string msg = "A problem has occurred with your quality evaluation.";
                            MessageBox.Show(msg, "Error");
                        }
                        else
                        {
                            if ((bool)dgvcBoxCell.Value == true)
                            {
                                int Index = this.dgvQualityStatement.Rows.IndexOf(dgvr);
                                processQualityEvaluation(Index);
                                anyProcessed = true;
                            }
                        }
                    }
                }
                if (anyProcessed)
                {
                    dREsults = new frmShowEvalResult();
                    dREsults.ReportEngine = _qe;
                    dREsults.Show(this);

                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                this.Cursor = Cursors.Default;
                this.statusText.Text = "Process concluded.";
            }
        }

        private Bentley.Interop.MicroStationDGN.Level getLevelbyName(string levelname)
        {
            Bentley.Interop.MicroStationDGN.Level retorno = null;
            Bentley.Interop.MicroStationDGN.Application Ustn = null;
            try
            {
                Ustn = ISOSpatialQuality.ComApp;
                if (Ustn != null)
                {
                    foreach (Bentley.Interop.MicroStationDGN.Level dLevel in Ustn.ActiveDesignFile.Leve
ls)
                    {
                        if (dLevel.Name.ToString().ToUpper() == levelname.ToUpper())
                        {
                            retorno = dLevel;
                            break;
                        }
                    }
                }
                return retorno;
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
                Ustn = null;
            }
        }

        private Bentley.Interop.MicroStationDGN.ElementScanCriteria setScanCriteria(SearchParametersQua
ntitativeAttribute searchP)
        {
            Bentley.Interop.MicroStationDGN.ElementScanCriteria retorno = new Bentley.Interop.MicroStat
ionDGN.ElementScanCriteriaClass();
            try {
                if (searchP != null)
                {
                    string layerName = searchP.levelName;
                    if (!string.IsNullOrEmpty(layerName))
                    {
                        Bentley.Interop.MicroStationDGN.Level dLevel = getLevelbyName(layerName);
                        if (dLevel != null)
                        {
                            retorno.ExcludeAllLevels();
                            retorno.IncludeLevel(dLevel);
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool elementIsOfColor(Bentley.Interop.MicroStationDGN.Element dElement, string color)
        {
            bool retorno = false;
            try
            {
                int dColor = -1;
                bool result = int.TryParse(color, out dColor);
                if (result)
                {
                    if (dElement.Color == dColor)
                    {
                        retorno = true;
                    }
                    else
                    {
                        if (dElement.Color == -1)
                        {
                            if (dElement.Level.ElementColor == dColor)
                            {
                                retorno = true;
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
```

```
        }

        private bool elementIsOfStyle(Bentley.Interop.MicroStationDGN.Element dElement, string lineStyl
e)
        {
            bool retorno = false;
            try
            {
                if (dElement.LineStyle.Name.ToString().ToUpper() == lineStyle.ToUpper())
                {
                    retorno = true;
                }
                else
                {
                    if (dElement.LineStyle.Name.ToString().ToUpper() == "BYLEVEL")
                    {
                        if (dElement.Level.ElementLineStyle.Name.ToString().ToUpper() == lineStyle.ToUp
per())
                        {
                            retorno = true;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private bool elementIsOfWeight(Bentley.Interop.MicroStationDGN.Element dElement, string lineWei
ght)
        {
            bool retorno = false;
            try
            {
                int lWeight = -1;
                bool result = int.TryParse(lineWeight, out lWeight);
                if (result)
                {
                    if (dElement.LineWeight == lWeight)
                    {
                        retorno = true;
                    }
                    else
                    {
                        if (dElement.LineWeight == -1)
                        {
                            if (dElement.Level.ElementLineWeight == lWeight)
                            {
                                retorno = true;
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private System.Collections.ArrayList getQAelements(
            Bentley.Interop.MicroStationDGN.ElementEnumerator ListOfElements,
            SearchParametersQuantitativeAttribute searchP,
            out int TotalElements,
            out int DetectedElements)
        {
            System.Collections.ArrayList retorno = null;
            TotalElements = 0;
```

```
            DetectedElements = 0;
            try {
                retorno = new System.Collections.ArrayList();
                while (ListOfElements.MoveNext())
                {
                    TotalElements++;
                    Bentley.Interop.MicroStationDGN.Element element = ListOfElements.Current;
                    if (!elementIsOfColor(element, searchP.lineColor))
                    {
                        DetectedElements++;
                        retorno.Add(element);
                    }
                    else
                    {
                        if (!elementIsOfStyle(element, searchP.styleNum))
                        {
                            DetectedElements++;
                            retorno.Add(element);
                        }
                        else
                        {
                            if (!elementIsOfWeight(element, searchP.lineWeight))
                            {
                                DetectedElements++;
                                retorno.Add(element);
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                TotalElements = 0;
                DetectedElements = 0;
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private Bentley.Interop.MicroStationDGN.ElementEnumerator getDGNElementsByLevel(string LevelNam
e)
        {
            Bentley.Interop.MicroStationDGN.ElementEnumerator ee = null;
            Bentley.Interop.MicroStationDGN.Application ustn = null;
            try {
                if (!string.IsNullOrEmpty(LevelName))
                {
                    Bentley.Interop.MicroStationDGN.Level dLevel = getLevelbyName(LevelName);
                    if (dLevel != null)
                    {
                        Bentley.Interop.MicroStationDGN.ElementScanCriteria sc = setSCByLevel(dLevel);
                        ustn = ISOSpatialQuality.ComApp;
                        if (ustn != null)
                        {
                            ee = ustn.ActiveModelReference.Scan(sc);
                        }
                    }
                }
                return ee;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                ee = null;
                ustn = null;
            }
        }

        private Bentley.Interop.MicroStationDGN.ElementScanCriteria setSCByLevel(Bentley.Interop.MicroS
```

```
tationDGN.Level dLevel)
        {
            Bentley.Interop.MicroStationDGN.ElementScanCriteria retorno = null;
            try {
                retorno = new Bentley.Interop.MicroStationDGN.ElementScanCriteriaClass();
                if (dLevel != null)
                {
                    retorno.ExcludeAllLevels();
                    retorno.IncludeLevel(dLevel);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool typeIsConformant(Bentley.Interop.MicroStationDGN.Element e, FeatureType ft)
        {
            //bool retorno = false;
            bool retorno = false;
            try {
                if (e != null)
                {
                    switch (e.Type)
                    {
                        case BIM.MsdElementType.Line:
                        case BIM.MsdElementType.ComplexString:
                        case BIM.MsdElementType.LineString:
                        case BIM.MsdElementType.MultiLine:
                            if (ft == FeatureType.linearFeature)
                            {
                                retorno = true;
                            }
                            if (ft == FeatureType.pointFeature)
                            {
                                if (e.AsLineElement().Length == 0)
                                {
                                    retorno = true;
                                }
                            }
                            break;
                        case BIM.MsdElementType.Shape:
                        case BIM.MsdElementType.ComplexShape:
                            if (ft == FeatureType.PolygonFeature)
                            {
                                retorno = true;
                            }
                            break;
                        case BIM.MsdElementType.CellHeader:
                        case BIM.MsdElementType.SharedCell:
                            if (ft == FeatureType.pointFeature)
                            {
                                retorno = true;
                            }
                            break;
                        default:
                            retorno = false;
                            break;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private System.Collections.ArrayList getNonConformantTypeElements(Bentley.Interop.MicroStationD
```

```
GN.ElementEnumerator ee, FeatureType fType, out int total)
        {
            System.Collections.ArrayList retorno = null;
            total = 0;
            try {
                retorno = new System.Collections.ArrayList();
                while (ee.MoveNext())
                {
                    Bentley.Interop.MicroStationDGN.Element el = ee.Current;
                    if (!typeIsConformant(el, fType))
                    {
                        retorno.Add(el);
                    }
                    total++;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool processFormatConsistency(SearchParametersFormatConsistency searchP, out int TotalE
lements, out int DetectedElements, out System.Collections.ArrayList detected)
        {
            bool retorno = false;
            Bentley.Interop.MicroStationDGN.ElementEnumerator listOfElements = null;
            detected = null;
            DetectedElements = 0;
            TotalElements = 0;
            try {
                string layername = searchP.LayerName;
                if (!string.IsNullOrEmpty(layername))
                {
                    listOfElements = getDGNElementsByLevel(layername);
                    if (listOfElements != null)
                    {
                        detected = getNonConformantTypeElements(listOfElements, searchP.Featuretype, ou
t TotalElements);
                        if (detected != null)
                        {
                            DetectedElements = detected.Count;
                            retorno = true;
                        }
                        else
                        {
                            DetectedElements = 0;
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                listOfElements = null;
            }
        }

        private List<string> domainValuesList(ListOfDomainValues lodv)
        {
            List<string> retorno = null;
            try {
                retorno = new List<string>();
                if (lodv != null)
                {
                    if (lodv.ListOfValues != null)
```

```
                {
                    foreach (string s in lodv.ListOfValues)
                    {
                        retorno.Add(s);
                    }
                }
            }
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally {
            retorno = null;
        }
    }

    private List<string> intervalValuesList(intervalClass ic)
    {
        List<string> retorno = null;
        try {
            retorno = new List<string>();
            if (ic != null)
            {
                int min = ic.min;
                int max = ic.max;
                for (int i = min; i <= max; i++)
                {
                    retorno.Add(i.ToString());
                }
            }
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally {
            retorno = null;
        }
    }

    private bool isValueInList(List<string> lista, string valor)
    {
        bool retorno = false;
        try {
            retorno = lista.Contains(valor);
            return retorno;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return false;
        }
    }

    private BIX.PropertyDef getFeatureProperty(BIX.feature f, string attributeName)
    {
        BIX.PropertyDef retorno = null;
        BIX.FeatureMgr fm = null;
        BIX.FeatureDef fd = null;
        try {
            fm = new BIX.FeatureMgr();
            fd = fm.GetFeatureDefinition(f.Name);
            if (fd != null)
            {
                retorno = fd.GetPropertyDefinitionByName(attributeName);
            }
            return null;
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
```

```
            finally {
                retorno = null;
                fm = null;
                fd = null;
            }
        }

        private System.Collections.ArrayList getDomainCheckIssues_ListOfValues(BIX.IFeatureEnumerator f
e, SearchParametersDomainConsistency spdc, out int TotalElements, out int DetectedElements)
        {
            System.Collections.ArrayList retorno = null;
            List<string> lista = new List<string>();
            TotalElements = 0;
            DetectedElements = 0;
            ListOfDomainValues lodv = null;
            try {
                retorno = new System.Collections.ArrayList();
                lodv = (ListOfDomainValues)spdc.DomainValues;
                if (lodv != null)
                {
                    lista = domainValuesList(lodv);
                    while (fe.MoveNext())
                    {
                        TotalElements++;
                        if (fe.Current.Name.ToUpper() == lodv.featureName.ToUpper())
                        {
                            if (!string.IsNullOrEmpty(spdc.DomainValues.attributeName))
                            {
                                string propertyValue = fe.Current.GetProperty(spdc.DomainValues.attribu
teName);
                                if (!isValueInList(lista, propertyValue))
                                {
                                    retorno.Add(fe.Current.Uuid);
                                    DetectedElements++;
                                }
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                TotalElements = 0;
                DetectedElements = 0;
                return null;
            }
            finally {
                retorno = null;
                lista = null;
                lodv = null;
            }
        }

        private bool isFieldNullOrEmpty(BIX.feature fe, string fieldName)
        {
            bool retorno = false;
            try {
                string valor = fe.GetProperty(fieldName);
                if (string.IsNullOrEmpty(valor))
                {
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
        }

        private detectionResultsCollection getFieldWithNullValues(BIX.IFeatureEnumerator fe, SearchPara
metersCompletenessRecord spcr)
        {
```

```
            detectionResultsCollection retorno = null;
            try {
                if (fe != null)
                {
                    if (spcr != null)
                    {
                        if (spcr.FeatureSchema != null)
                        {
                            int[] detected = new int[spcr.FeatureSchema.FieldCollection.Count];
                            int total = 0;
                            while (fe.MoveNext())
                            {
                                BIX.feature fc = fe.Current;
                                for (int i = 0; i < spcr.FeatureSchema.FieldCollection.Count; i++)
                                {
                                    if (isFieldNullOrEmpty(fc, spcr.FeatureSchema.FieldCollection[i].Na
me))
                                    {
                                        detected[i]++;
                                    }

                                }
                                total++;
                            }
                            retorno = new detectionResultsCollection();
                            for (int i = 0; i < spcr.FeatureSchema.FieldCollection.Count; i++)
                            {
                                detectionResults dr = new detectionResults();
                                dr.detectedElements = detected[i];
                                dr.totalElements = total;
                                dr.detectedIDs = null;
                                retorno.addDetectionResults(dr);
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private detectionResults getFieldWithNullValues(System.Collections.ArrayList fe, string fieldNa
me)
        {
            int totalEl = 0;
            int detectEl = 0;
            detectionResults retorno = null;
            System.Collections.ArrayList detectedID = null;
            try {
                if (fe != null)
                {
                    retorno = new detectionResults();
                    detectedID = new System.Collections.ArrayList();
                    for (int i = 0; i < fe.Count; i++)
                    {
                        BIX.featureClass fc = (BIX.featureClass)fe[i];
                        string propertyValue = fc.GetProperty(fieldName);
                        if (string.IsNullOrEmpty(propertyValue))
                        {
                            //detectedID.Add(fc.Uuid);
                            detectEl++;
                        }
                        totalEl++;
                        fc = null;
                    }
                }
                GC.Collect();
                if (retorno != null)
```

```
                {
                    retorno.totalElements = totalEl;
                    retorno.detectedElements = detectEl;
                    retorno.detectedIDs = detectedID;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
                detectedID = null;
            }
        }

        private BIX.IFeatureEnumerator getFeatures(string featureName)
        {
            BIX.IFeatureEnumerator retorno = null;
            BIX.FeatureMgr fm = null;
            try {
                fm = new BIX.FeatureMgr();
                retorno = fm.GetSessionFeaturesList();
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
                fm = null;
            }
        }

        private void SearchElements_CompletenessRecord(QualityReport qr, int Index)
        {
            detectionResultsCollection drc = null;
            SearchParametersCompletenessRecord spcr = null;
            //System.Collections.ArrayList features = null;
            BIX.IFeatureEnumerator features = null;
            try {
                if (qr != null)
                {
                    if (qr.searchParams != null)
                    {
                        drc = new detectionResultsCollection();
                        spcr = (SearchParametersCompletenessRecord)qr.searchParams;
                        features = getFeatures(spcr.FeatureName);
                        drc = getFieldWithNullValues(features, spcr);
                        this._qe.QualityCollection[Index].DetectedElementsCol = drc;
                    }
                }
                //this._qe.QualityCollection[Index].DetectedElementsCol = drc;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                drc = null;
                spcr = null;
                features = null;
            }
        }

        private System.Collections.ArrayList getDomainCheckIssues_IntervalValues(BIX.IFeatureEnumerator
fe, SearchParametersDomainConsistency spdc, out int TotalElements, out int DetectedElements)
        {
            List<string> lista = null;
            System.Collections.ArrayList retorno = null;
            intervalClass ic = null;
            TotalElements = 0;
            DetectedElements = 0;
```

```
            try {
                retorno = new System.Collections.ArrayList();
                if (spdc != null)
                {
                    ic = (intervalClass)spdc.DomainValues;
                    lista = intervalValuesList(ic);
                    while (fe.MoveNext())
                    {
                        TotalElements++;
                        if (fe.Current.Name.ToUpper() == ic.featureName.ToUpper())
                        {
                            if (!string.IsNullOrEmpty(ic.attributeName))
                            {
                                string propertyValue = fe.Current.GetProperty(ic.attributeName);
                                if (!isValueInList(lista, propertyValue))
                                {
                                    retorno.Add(fe.Current.Uuid);
                                    DetectedElements++;
                                }
                            }
                        }
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                lista = null;
                retorno = null;
                ic = null;
            }
        }

        private System.Collections.ArrayList getDomainCheckIssues(BIX.IFeatureEnumerator fe, SearchPara
metersDomainConsistency spdc, out int TotalElements, out int DetectedElements)
        {
            System.Collections.ArrayList retorno = null;
            TotalElements = 0;
            DetectedElements = 0;
            try {
                retorno = new System.Collections.ArrayList();
                string featureName = string.Empty;
                string attributeName = string.Empty;
                if (spdc != null)
                {
                    if (spdc.DomainValues != null)
                    {
                        switch (spdc.DomainValues.domainType)
                        {
                            case DomainType.undefined:
                                retorno = new System.Collections.ArrayList();
                                TotalElements = 0;
                                DetectedElements = 0;
                                break;
                            case DomainType.ListOfValues:
                                retorno = getDomainCheckIssues_ListOfValues(fe, spdc, out TotalElements
, out DetectedElements);
                                break;
                            case DomainType.ValueInterval:
                                retorno = getDomainCheckIssues_IntervalValues(fe, spdc, out TotalElemen
ts, out DetectedElements);
                                break;
                            default:
                                retorno = new System.Collections.ArrayList();
                                TotalElements = 0;
                                DetectedElements = 0;
                                break;
                        }
                    }
                }
                return retorno;
```

```
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private bool processDomainCheck(SearchParametersDomainConsistency spdc, out int TotalElements,
out int DetectedElements, out System.Collections.ArrayList detected)
        {
            BIX.IFeatureEnumerator fe = null;
            BIX.FeatureMgr fm = null;
            detected = null;
            bool retorno = false;
            TotalElements = 0;
            DetectedElements = 0;
            detected = null;
            try {
                fm = new BIX.FeatureMgr();
                fe = fm.GetSessionFeaturesList();
                detected = getDomainCheckIssues(fe, spdc, out TotalElements, out DetectedElements);
                if (detected != null)
                {
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                fe = null;
                fm = null;
            }
        }

        private bool processQuantitativeAttribute(SearchParametersQuantitativeAttribute searchP, out in
t TotalElements, out int DetectedElements, out System.Collections.ArrayList detectedCollection)
        {
            bool retorno = false;
            Bentley.Interop.MicroStationDGN.Application Ustn = null;
            Bentley.Interop.MicroStationDGN.ElementScanCriteria sCriteria = null;
            Bentley.Interop.MicroStationDGN.ElementEnumerator ListOfElements = null;
            System.Collections.ArrayList Detected = null;
            TotalElements = 0;
            DetectedElements = 0;
            detectedCollection = null;
            try {
                Ustn = ISOSpatialQuality.ComApp;
                sCriteria = setScanCriteria(searchP);
                if (Ustn != null)
                {
                    ListOfElements = Ustn.ActiveModelReference.Scan(sCriteria);
                    Detected = getQAelements(ListOfElements, searchP, out TotalElements, out DetectedEl
ements);
                    detectedCollection = Detected;
                    retorno = true;
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return false;
            }
            finally {
                Ustn = null;
                sCriteria = null;
                ListOfElements = null;
            }
        }
```

```
        private void SearchElements_DomainCheck(QualityReport qr, int Index)
        {
            SearchParametersDomainConsistency spdc = null;
            System.Collections.ArrayList retorno = null;
            detectionResults dr = null;
            try {
                spdc = (SearchParametersDomainConsistency)qr.searchParams;
                int totalEle = 0;
                int detectEle = 0;
                bool result = processDomainCheck(spdc, out totalEle, out detectEle, out retorno);
                if (result)
                {
                    dr = new detectionResults();
                    dr.detectedElements = detectEle;
                    dr.detectedIDs = retorno;
                    dr.totalElements = totalEle;
                    this._qe.QualityCollection[Index].DetectedElements = dr;
                }
            }
            catch (Exception ex) { MessageBox.Show(ex.Message.ToString(), "Error"); }
            finally {
                spdc = null;
                retorno = null;
                dr = null;
            }
        }

        private void SearchElements_formatConsistency(QualityReport qr, int Index)
        {
            SearchParametersFormatConsistency spfc = null;
            System.Collections.ArrayList retorno = null;
            detectionResults dr = null;
            try {
                spfc = (SearchParametersFormatConsistency)qr.searchParams;
                int totalEl = 0;
                int detectEl = 0;
                bool result = processFormatConsistency(spfc, out totalEl, out detectEl, out retorno);
                if (result)
                {
                    dr = new detectionResults();
                    dr.totalElements = totalEl;
                    dr.detectedElements = detectEl;
                    dr.detectedIDs = retorno;
                    this._qe.QualityCollection[Index].DetectedElements = dr;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                spfc = null;
                retorno = null;
                dr = null;
            }
        }

        private void SearchElements_Completeness(QualityReport qr, int Index)
        {
            SearchParametersCompleteness spc = null;
            try {
                spc = (SearchParametersCompleteness)qr.searchParams;
                switch (spc.completenessAssessment)
                {
                    case CompletenessAssessment.CompletenessRecord:
                        SearchElements_CompletenessRecord(qr, Index);
                        break;
                    case CompletenessAssessment.CompletenessSchema:
                        break;
                    case CompletenessAssessment.undefined:
                        break;
                    default:
                        break;
                }
```

```
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                spc = null;
            }
        }

        private void SearchElements(QualityReport qr, int Index)
        {
            SearchParametersQuantitativeAttribute spqa = null;
            System.Collections.ArrayList retorno = null;
            detectionResults dr = null;
            try {
                spqa = (SearchParametersQuantitativeAttribute)qr.searchParams;
                int totalElements = 0;
                int detectedElements = 0;
                bool result = processQuantitativeAttribute(spqa, out totalElements, out detectedElement
s, out retorno);
                if (result)
                {
                    dr = new detectionResults();
                    dr.totalElements = totalElements;
                    dr.detectedElements = detectedElements;
                    dr.detectedIDs = retorno;
                    this._qe.QualityCollection[Index].DetectedElements = dr;
                }
                //return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                //return null;
            }
            finally {
                spqa = null;
                retorno = null;
                dr = null;
            }
        }

        private void SearchElements_phisicalFileFormat(QualityReport qr, int Index)
        {
            try {

            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally { }
        }

        private void processQualityEvaluation(int Index)
        {
            QualityReport qr = null;

            try {
                qr = this._qe.QualityCollection[Index];

                if (qr != null)
                {
                    this.statusText.Text = "Processing " + qr.nameOfMeasure + ", please wait...";
                    Application.DoEvents();
                    switch (qr.subElement)
                    {
                        case QualitySubelement.DQ_AbsoluteExternalPositionalAccuracy:
                            break;
                        case QualitySubelement.DQ_AccuracyOfATimeMeasurement:
                            break;
                        case QualitySubelement.DQ_CompletenessComission:
                        case QualitySubelement.DQ_CompletenessOmission:
                            SearchElements_Completeness(qr, Index);
                            break;
                        case QualitySubelement.DQ_ConceptualConsistency:
```

```
                            SearchElements_formatConsistency(qr, Index);
                            break;
                        case QualitySubelement.DQ_DomainConsistency:
                            SearchElements_DomainCheck(qr, Index);
                            break;
                        case QualitySubelement.DQ_FormatConsistency:
                            break;
                        case QualitySubelement.DQ_GriddedDataPositionalAccuracy:
                            break;
                        case QualitySubelement.DQ_NonQuantitativeAttributeAccuracy:
                            SearchElements(qr, Index);
                            break;
                        case QualitySubelement.DQ_QuantitativeAttributeAccuracy:
                            //SearchElements(qr, Index);
                            break;
                        case QualitySubelement.DQ_RelativeInternalPositionalAccuracy:
                            break;
                        case QualitySubelement.DQ_TemporalConsistency:
                            break;
                        case QualitySubelement.DQ_TemporalValidity:
                            break;
                        case QualitySubelement.DQ_ThematicClassificationCorrectness:
                            break;
                        case QualitySubelement.DQ_TopologicalConsistency:
                            break;
                        default:
                            break;
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                qr = null;
            }
        }

        private void dgvQualityStatement_CellValueNeeded(object sender, DataGridViewCellValueEventArgs
e)
        {
            QualityReport qr = null;
            try {
                int rIndex = -1;
                rIndex = e.RowIndex;
                if (rIndex > -1)
                {
                    qr = this._qe.QualityCollection[rIndex];
                    QualityStatus qs = qr.getUserReportStatus();

                    switch (qs)
                    {
                        case QualityStatus.accepted:
                            e.Value = appResource.ACCEPTED;
                            break;
                        case QualityStatus.rejected:
                            e.Value = appResource.REJECTED;
                            break;
                        case QualityStatus.undefined:
                            e.Value = appResource.Blank16;
                            break;
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                qr = null;
            }
        }

        private void cmdGetLineage_Click(object sender, EventArgs e)
        {
```

```
            frmLineage fl = null;
            try {
                fl = new frmLineage();
                fl.Lineage = this._lineage;
                if (fl.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                {
                    this._lineage = fl.Lineage;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                fl.Close();
                fl = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Utilities;
using BIM = Bentley.Interop.MicroStationDGN;


namespace ISOSpatialQuality.Forms
{
    public partial class frmQuantitativeAttribute : Form
    {
        private QualityReport _qr = null;

        public QualityReport QualityReport
        {
            get { return _qr; }
            set { _qr = value; }
        }

        private void loadSearchInfo()
        {
            try {
                if (_qr != null)
                {
                    loadQualityReportInfo();
                }
            }
            catch (Exception ex) {
                throw new ArgumentException("Error", ex);
            }
        }

        private void loadQualityReportInfo()
        {
            try
            {
                this.lblDQ_ELEMENT.Text = _qr.DQ_Element;
                this.lblDQ_SUBELEMENT.Text = _qr.DQ_Subelement;
                this.txtEvalDescription.Text = _qr.DQ_EvalMethodDesc;
                loadSearchParameters(_qr.searchParams);
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }
```

```
        private void loadSearchParameters(SearchParameters sp)
        {
            try {
                if (sp != null)
                {
                    SearchParametersQuantitativeAttribute spqa = (SearchParametersQuantitativeAttribute
)sp;
                    if (spqa != null)
                    {
                        this.cboLevel.Text = spqa.levelName;
                        this.cboLineStyle.Text = spqa.styleNum;
                        this.txtColorNum.Text = spqa.lineColor;
                        this.txtLineWeight.Text = spqa.lineWeight;
                    }
                    else
                    {
                        this.cboLevel.SelectedIndex = 0;
                        this.cboLineStyle.SelectedIndex = 0;
                        this.txtColorNum.Text = string.Empty;
                        this.txtLineWeight.Text = string.Empty;
                    }
                }
                else
                {
                    this.cboLevel.SelectedIndex = 0;
                    this.cboLineStyle.SelectedIndex = 0;
                    this.txtColorNum.Text = string.Empty;
                    this.txtLineWeight.Text = string.Empty;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        public frmQuantitativeAttribute()
        {
            InitializeComponent();
        }

        private void frmQuantitativeAttribute_ResizeEnd(object sender, EventArgs e)
        {
            this.Width=350;
            this.Height=365;
        }

        private SearchParametersQuantitativeAttribute setSearchParameters()
        {
            SearchParametersQuantitativeAttribute retorno = null;
            try {
                retorno = new SearchParametersQuantitativeAttribute();
                retorno.levelName = this.cboLevel.Text;
                retorno.lineColor = this.txtColorNum.Text;
                retorno.lineWeight = this.txtLineWeight.Text;
                retorno.styleNum = this.cboLineStyle.Text;

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            SearchParametersQuantitativeAttribute parameters = null;
            try {
                parameters = setSearchParameters();
                if (parameters != null)
```

```
                {
                    this._qr.searchParams = parameters;
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
                else
                {
                    this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                parameters = null;
            }
        }

        private System.Collections.ArrayList getDGNLineStyles()
        {
            System.Collections.ArrayList retorno = null;
            Bentley.Interop.MicroStationDGN.Application dMstn = null;
            try
            {
                dMstn = ISOSpatialQuality.ComApp;
                if (dMstn.ActiveDesignFile.LineStyles.Count > 0)
                {
                    retorno = new System.Collections.ArrayList();
                    foreach (BIM.LineStyle lStyle in dMstn.ActiveDesignFile.LineStyles)
                    {
                        string lsName = lStyle.Name.ToString();
                        retorno.Add(lsName);
                    }
                }
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                dMstn = null;
                retorno = null;
            }
        }

        private System.Collections.ArrayList getDGNLevels()
        {
            System.Collections.ArrayList dLevels = null;
            Bentley.Interop.MicroStationDGN.Application dMstn;
            try
            {
                dMstn = ISOSpatialQuality.ComApp;

                if (dMstn.ActiveDesignFile.Levels.Count > 0)
                {
                    dMstn = ISOSpatialQuality.ComApp;

                    dLevels = new System.Collections.ArrayList();

                    foreach (BIM.Level dLevel in dMstn.ActiveDesignFile.Levels)
                    {
                        string LevelName = dLevel.Name.ToString();
                        dLevels.Add(LevelName);
                    }
                }

                return dLevels;
            }
            catch (Exception ex)
            {
```

```
            MessageBox.Show(ex.Message.ToString(), "Error");
            return null;
        }
        finally
        {
            dMstn = null;
            dLevels = null;
        }
    }

    private void loadLevelsinComboBox(System.Collections.ArrayList levels, ComboBox dCombo)
    {
        try
        {
            dCombo.Items.Clear();
            foreach (string lName in levels)
            {
                dCombo.Items.Add(lName);
            }
            dCombo.Items.Insert(0, "Select a level/layer name...");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Warning");
        }
    }

    private void loadLineStylesinComboBox(System.Collections.ArrayList lStyles, ComboBox dcombo)
    {
        try
        {
            dcombo.Items.Clear();
            foreach (string lsName in lStyles)
            {
                dcombo.Items.Add(lsName);
            }
            dcombo.Items.Insert(0, "Select a line style...");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Warning");
        }
    }

    private void loadCombos()
    {
        System.Collections.ArrayList linestyles = null;
        System.Collections.ArrayList levelnames = null;
        try
        {
            linestyles = getDGNLineStyles();
            levelnames = getDGNLevels();
            loadLevelsinComboBox(levelnames, this.cboLevel);
            loadLineStylesinComboBox(linestyles, this.cboLineStyle);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Warning");
        }
        finally
        {
            linestyles = null;
            levelnames = null;
        }
    }

    private void frmQuantitativeAttribute_Load(object sender, EventArgs e)
    {
        loadCombos();
        loadSearchInfo();
    }

    private void cmdCancel_Click(object sender, EventArgs e)
    {
```

```
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmShowEvalResult : Form
    {
        private QualityReportEngine _qre = null;
        private DBUtils _dbUtil = null;

        private string getMeasureString(int num)
        {
            string retorno = string.Empty;
            try {
                DataTable dados = getMeasureType();

                DataRow[] rows = dados.Select("ID_MEASURE = " + num.ToString());

                if (rows.Length > 0)
                {
                    retorno = rows[0]["MEASURE_DESCRIPTION"].ToString();
                }

                //dados.Select("ID_MEASURE = " + num.ToString());
                //if (dados.Rows.Count > 0)
                //{
                //    retorno = dados.Rows[0]["MEASURE_DESCRIPTION"].ToString();
                //}
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return string.Empty;
            }
        }

        public QualityReportEngine ReportEngine { get { return _qre; } set { _qre = value; } }

        private DataTable getMeasureType()
        {
            DataTable retorno = null;
            try
            {
                string sql = "select ID_MEASURE, MEASURE_DESCRIPTION FROM DQ_MEASURES";
                retorno = _dbUtil.getItemsFromDatabase(sql);
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }
```

```
        private void loadTree()
        {
            try {
                foreach (QualityReport qr in _qre.QualityCollection)
                {
                    TreeNode node = new TreeNode();
                    node.Text = qr.nameOfMeasure;
                    node.Tag = "ID_" + this._qre.QualityCollection.IndexOf(qr);

                    TreeNode dElement = new TreeNode();
                    dElement.Text = qr.DQ_Element;

                    TreeNode dSubElement = new TreeNode();
                    dSubElement.Text = qr.DQ_Subelement;

                    node.Nodes.Add(dElement);
                    node.Nodes.Add(dSubElement);

                    this.tvQReport.Nodes.Add(node);
                    node = null;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(),"Error");
            }
        }

        public frmShowEvalResult()
        {
            InitializeComponent();
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            this.Hide();
            this.Owner.Show();
        }

        private void frmShowEvalResult_Load(object sender, EventArgs e)
        {
            //this._qre = new QualityReportEngine();
            _dbUtil = new DBUtils();
            this.loadTree();
            //loadNodeInfo(null);
            loadResultInformation(null);
        }

        private void tvQReport_NodeMouseClick(object sender, TreeNodeMouseClickEventArgs e)
        {
            TreeNode SelectedNode = null;
            try {
                SelectedNode = e.Node;
                //loadNodeInfo(SelectedNode);
                //loadResultInformation(SelectedNode);
                showQualityResults(SelectedNode);
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                SelectedNode = null;
            }
        }

        private string getOperatorString(int valor)
        {
            string retorno = string.Empty;
            try {
                if (valor > 0)
                {
                    switch(valor)
                    {
                        case 1:
```

```
                            retorno="=";
                            break;
                        case 2:
                            retorno = ">";
                            break;
                        case 3:
                            retorno = ">=";
                            break;
                        case 4:
                            retorno = "<";
                            break;
                        case 5:
                            retorno = "<=";
                            break;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return string.Empty;
            }
        }

        private string getEvalResult(string MeasureType, QualityReport qr)
        {
            string retorno = string.Empty;
            try {
                if (qr.DetectedElements != null)
                {
                    detectionResults dr = qr.DetectedElements;
                    int valorcoletado = dr.detectedElements;
                    int totalElements = dr.totalElements;

                    switch (MeasureType.ToUpper())
                    {
                        case "COUNT":
                            retorno = valorcoletado.ToString();
                            break;
                        case "RATIO":
                            retorno = (valorcoletado / totalElements).ToString();
                            break;
                        case "PERCENTAGE":
                            retorno = System.Math.Round((((double)valorcoletado / (double)totalElements
) * 100),4).ToString() + "%";
                            break;
                        case "RMSE":
                            break;
                        case "STANDARD DEVIATION":
                            break;
                        case "PASS_FAIL":
                            retorno = valorcoletado.ToString();
                            break;
                    }
                 }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return string.Empty;
            }
            finally { }
        }

        private void showDetectedQualityInfo_QuantityInformation(QualityReport qr)
        {
            try {
                if (qr.DetectedElements != null)
                {
                    ExpectedQuality eq = (ExpectedQuality)qr.expectedQuality;
                    if (eq != null)
                    {
                        this.lblMeasTypeER.Text = getMeasureString(eq.MeasureType);
                        this.lblValueCollected.Text = getEvalResult(this.lblMeasTypeER.Text, qr);
```

```
                    string user_result = qr.getUserReportStatus().ToString();
                    string provider_result = qr.getProviderReportStatus().ToString();
                    this.PROVIDER_QA.Text = provider_result;
                    this.USER_QA.Text = user_result;
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void showExpectedQualityInfo_QuantityInformation(QualityReport qr)
    {
        try {
            if (qr != null)
            {
                ExpectedQuality eq = (ExpectedQuality)qr.expectedQuality;
                if (eq != null)
                {
                    this.lblApprovedReproved.Text = eq.AcceptOption.ToString();
                    string measureType = getMeasureString(eq.MeasureType);
                    this.lblMEASUREMENTTYPE.Text = measureType;
                    this.lblCompare.Text = getOperatorString(eq.COperator.GetHashCode());
                    this.lblValue.Text = eq.Value;
                    this.lblValueUnit.Text = eq.ValueUnit;
                }
                else {
                    this.lblApprovedReproved.Text = string.Empty;
                    string measureType = string.Empty;
                    this.lblMEASUREMENTTYPE.Text = string.Empty;
                    this.lblCompare.Text = string.Empty;
                    this.lblValue.Text = string.Empty;
                    this.lblValueUnit.Text = string.Empty;
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void cleanUpFields()
    {
        this.lblDQ_ELEMENT.Text = string.Empty;
        this.lblDQ_SUBELEMENT.Text = string.Empty;
        this.txtDQ_MEASUREDESCRIPTION.Text = string.Empty;
        this.txtEvalDescription.Text = string.Empty;
        this.lblApprovedReproved.Text = string.Empty;
        this.lblMEASUREMENTTYPE.Text = string.Empty;
        this.lblCompare.Text = string.Empty;
        this.lblValue.Text = string.Empty;
        this.lblValueUnit.Text = string.Empty;
        this.lblMeasTypeER.Text = string.Empty;
        this.lblValueCollected.Text = string.Empty;
        this.PROVIDER_QA.Text = string.Empty;
        this.USER_QA.Text = string.Empty;
    }

    private int getNodeID(TreeNode node)
    {
        int retorno = -1;
        try {
            if (node.Parent == null)
            {
                string idText = node.Tag.ToString();
                string[] info = idText.Split(new char[] { '_' });
                string id = info[1];
                int aux = -1;
                bool result = int.TryParse(id, out aux);
                if (result)
                {
                    retorno = aux;
                }
```

```
                }
                else
                {
                    retorno = getNodeID(node.Parent);
                }
                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void showExpectedQualityInfo_CompletenessSchema(QualityReport qr)
        {
            ExpectedCompletenessSchema ecs = null;
            try {
                ecs = (ExpectedCompletenessSchema)qr.expectedQuality;
                if (ecs != null)
                {
                    this.lblApprovedReproved.Text = ecs.acceptOption.ToString();
                    //string measureType = getMeasureString(eq.MeasureType);
                    string measureType = qr.subElement.ToString();
                    this.lblMEASUREMENTTYPE.Text = measureType;
                    this.lblCompare.Text = string.Empty;
                    this.lblValue.Text = "exists";
                    this.lblValueUnit.Text = string.Empty;
                    this.PROVIDER_QA.Text = string.Empty;
                    this.USER_QA.Text = qr.getUserReportStatus().ToString();
                }
                else
                {
                    this.lblApprovedReproved.Text = string.Empty;
                    string measureType = string.Empty;
                    this.lblMEASUREMENTTYPE.Text = string.Empty;
                    this.lblCompare.Text = string.Empty;
                    this.lblValue.Text = string.Empty;
                    this.lblValueUnit.Text = string.Empty;
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                ecs = null;
            }
        }

        private void showExpectedQualityInfo_RecordInformation(QualityReport qr)
        {
            try {
                if (qr != null)
                {
                    if (qr.expectedQuality != null)
                    {
                        ExpectedCompletenessRecord ecr = (ExpectedCompletenessRecord)qr.expectedQuality
;

                        if (ecr != null)
                        {
                            this.lblApprovedReproved.Text = ecr.acceptOption.ToString();
                            string measureType = getMeasureString(ecr.MeasureType);
                            this.lblMEASUREMENTTYPE.Text = measureType;
                            this.lblCompare.Text = getOperatorString(ecr.COperator.GetHashCode());
                            this.lblValue.Text = ecr.Value;
                            this.lblValueUnit.Text = ecr.ValueUnit;
                        }
                        else
                        {
                            this.lblApprovedReproved.Text = string.Empty;
                            string measureType = string.Empty;
                            this.lblMEASUREMENTTYPE.Text = string.Empty;
                            this.lblCompare.Text = string.Empty;
                            this.lblValue.Text = string.Empty;
```

```
                        this.lblValueUnit.Text = string.Empty;
                    }

                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void showDetectedQualityInfo_RecordInformation(QualityReport qr)
    {
        int index = -1;
        int higher = 0;
        try {
            if (qr != null)
            {
                if (qr.DetectedElementsCol != null)
                {
                    for (int i = 0; i < qr.DetectedElementsCol.Count; i++)
                    {
                        detectionResults dr = qr.DetectedElementsCol[i];
                        if (dr.detectedElements > higher)
                        {
                            higher = dr.detectedElements;
                            index = i;
                        }
                    }
                }
            }
            if (index > -1)
            {
                detectionResults dr = qr.DetectedElementsCol[index];
                if (dr != null)
                {
                    qr.DetectedElements = dr;
                }
                ExpectedCompletenessRecord ecr = (ExpectedCompletenessRecord)qr.expectedQuality;
                if (dr != null)
                {
                    if (ecr != null)
                    {
                        this.lblMeasTypeER.Text = getMeasureString(ecr.MeasureType);
                        this.lblValueCollected.Text = getEvalResult(this.lblMeasTypeER.Text, qr);
                        string user_result = qr.getUserReportStatus().ToString();
                        string provider_result = qr.getProviderReportStatus().ToString();
                        //this.PROVIDER_QA.Text = provider_result;
                        this.USER_QA.Text = user_result;
                    }
                }
            }
        }
        catch (Exception ex) { }
        finally { }
    }

    private void showExpectedQualityInfo_CompletenessConsistency(QualityReport qr)
    {
        try {
            if (qr != null)
            {
                ExpectedCompleteness eq = (ExpectedCompleteness)qr.expectedQuality;
                if (eq != null)
                {
                    switch (eq.completenessAssessment)
                    {
                        case CompletenessAssessment.CompletenessSchema:
                            showExpectedQualityInfo_CompletenessSchema(qr);
                            break;
                        case CompletenessAssessment.CompletenessRecord:
                            showExpectedQualityInfo_RecordInformation(qr);
                            showDetectedQualityInfo_RecordInformation(qr);
                            break;
```

```
                        case CompletenessAssessment.undefined:
                            break;
                        default:
                            break;
                    }
                }
                else
                {
                    this.lblApprovedReproved.Text = string.Empty;
                    string measureType = string.Empty;
                    this.lblMEASUREMENTTYPE.Text = string.Empty;
                    this.lblCompare.Text = string.Empty;
                    this.lblValue.Text = string.Empty;
                    this.lblValueUnit.Text = string.Empty;
                }
            }
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void showExpectedQualityInfo_FormatConsistency(QualityReport qr)
    {
        ExpectedFormatConsistencyCAD efcc = null;
        try {
            efcc = (ExpectedFormatConsistencyCAD)qr.expectedQuality;
            if (efcc != null)
            {
                this.lblApprovedReproved.Text = efcc.acceptOption.ToString();
                //string measureType = getMeasureString(eq.MeasureType);
                string measureType = "File format";
                this.lblMEASUREMENTTYPE.Text = measureType;
                this.lblCompare.Text = "is";
                this.lblValue.Text = efcc.fileFormat.ToString();
                this.lblValueUnit.Text = string.Empty;
                this.PROVIDER_QA.Text = string.Empty;
                this.USER_QA.Text = qr.getUserReportStatus().ToString();
            }
            //this.USER_QA.Text = qr.getUserReportStatus().ToString();
        }
        catch (Exception ex) {
            MessageBox.Show(ex.Message.ToString(), "Error");
        }
    }

    private void showQualityResults(TreeNode node)
    {
        try {
            cleanUpFields();
            int nodeID = getNodeID(node);
            if (nodeID > -1)
            {
                QualityReport qr = this._qre.QualityCollection[nodeID];
                if (qr != null)
                {
                    this.lblDQ_ELEMENT.Text = qr.DQ_Element;
                    this.lblDQ_SUBELEMENT.Text = qr.DQ_Subelement;
                    this.txtDQ_MEASUREDESCRIPTION.Text = qr.DQ_MeasureDesc;
                    this.txtEvalDescription.Text = qr.DQ_EvalMethodDesc;

                    switch (qr.subElement)
                    {
                        case QualitySubelement.DQ_AbsoluteExternalPositionalAccuracy:
                            break;
                        case QualitySubelement.DQ_AccuracyOfATimeMeasurement:
                            break;
                        case QualitySubelement.DQ_CompletenessComission:
                        case QualitySubelement.DQ_CompletenessOmission:
                            showExpectedQualityInfo_CompletenessConsistency(qr);
                            break;
                        case QualitySubelement.DQ_ConceptualConsistency:
                            showExpectedQualityInfo_QuantityInformation(qr);
                            showDetectedQualityInfo_QuantityInformation(qr);
```

```
                                break;
                            case QualitySubelement.DQ_DomainConsistency:
                                showExpectedQualityInfo_QuantityInformation(qr);
                                showDetectedQualityInfo_QuantityInformation(qr);
                                break;
                            case QualitySubelement.DQ_FormatConsistency:
                                showExpectedQualityInfo_FormatConsistency(qr);
                                break;
                            case QualitySubelement.DQ_GriddedDataPositionalAccuracy:
                                break;
                            case QualitySubelement.DQ_NonQuantitativeAttributeAccuracy:
                                showExpectedQualityInfo_QuantityInformation(qr);
                                showDetectedQualityInfo_QuantityInformation(qr);
                                break;
                            case QualitySubelement.DQ_QuantitativeAttributeAccuracy:
                                //showExpectedQualityInfo_QuantityInformation(qr);
                                //showDetectedQualityInfo_QuantityInformation(qr);
                                break;
                            case QualitySubelement.DQ_RelativeInternalPositionalAccuracy:
                                break;
                            case QualitySubelement.DQ_TemporalConsistency:
                                break;
                            case QualitySubelement.DQ_TemporalValidity:
                                break;
                            case QualitySubelement.DQ_ThematicClassificationCorrectness:
                                break;
                            case QualitySubelement.DQ_TopologicalConsistency:
                                break;
                            case QualitySubelement.undefined:
                                break;
                            default:
                                break;
                        }
                    }
                    else
                    {
                        cleanUpFields();
                    }
                }
                else
                {
                    cleanUpFields();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadResultInformation(TreeNode node)
        {
            this.lblDQ_ELEMENT.Text = string.Empty;
            this.lblDQ_SUBELEMENT.Text = string.Empty;
            this.txtDQ_MEASUREDESCRIPTION.Text = string.Empty;
            this.txtEvalDescription.Text = string.Empty;
            this.lblApprovedReproved.Text = string.Empty;
            this.lblMEASUREMENTTYPE.Text = string.Empty;
            this.lblCompare.Text = string.Empty;
            this.lblValue.Text = string.Empty;
            this.lblValueUnit.Text = string.Empty;
            this.lblMeasTypeER.Text = string.Empty;
            this.lblValueCollected.Text = string.Empty;
            this.PROVIDER_QA.Text = string.Empty;
            this.USER_QA.Text = string.Empty;
            try {
                if (node != null)
                {
                    if (node.Parent == null)
                    {
                        string idText = node.Tag.ToString();
                        string[] info = idText.Split(new char[] { '_' });
                        string id = info[1];
                        int index = -1;
                        bool result = int.TryParse(id, out index);
```

```
                        if (result)
                        {
                            QualityReport qr = this._qre.QualityCollection[index];
                            this.lblDQ_ELEMENT.Text = qr.DQ_Element;
                            this.lblDQ_SUBELEMENT.Text = qr.DQ_Subelement;
                            this.txtDQ_MEASUREDESCRIPTION.Text = qr.DQ_MeasureDesc;
                            this.txtEvalDescription.Text = qr.DQ_EvalMethodDesc;
                            switch (qr.DQ_Subelement.ToUpper())
                            {
                                case "FORMAT CONSISTENCY":
                                    break;
                                default:
                                    showExpectedQualityInfo_QuantityInformation(qr);
                                    break;
                            }
                        }
                    }
                    else
                    {
                        loadResultInformation(node.Parent);
                    }
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }


        private void cmdElementNavigation_Click(object sender, EventArgs e)
        {

        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmUserExpectedCompletenessRecord : Form
    {
        private DBUtils _dbUtil = null;
        private QualityReport _qr = null;

        public QualityReport qualityReport { get { return _qr; } set { _qr = value; } }

        public frmUserExpectedCompletenessRecord()
        {
            InitializeComponent();
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }

        private void frmUserExpectedCompletenessRecord_Load(object sender, EventArgs e)
        {
            this._dbUtil = new DBUtils();
            loadMeasureType(this.cboDQ_MEASUREDESC);
```

```
            this.cboOperator.SelectedIndex = 0;
            this.cboPassFail.SelectedIndex = 0;
            loadExistingInfo();
        }

        private void loadMeasureType(ComboBox dcombo)
        {
            try
            {
                DataTable EvalTypes = getMeasureType();
                if (EvalTypes != null)
                {
                    DataRow dRow = EvalTypes.NewRow();
                    dRow[0] = 0;
                    dRow[1] = "Select a Measurement Type...";

                    if (EvalTypes.Rows.Count > 0)
                    {
                        EvalTypes.Rows.InsertAt(dRow, 0);
                    }
                    else
                    {
                        EvalTypes.Rows.Add(dRow);
                    }
                    dcombo.DisplayMember = "MEASURE_DESCRIPTION";
                    dcombo.ValueMember = "ID_MEASURE";
                    dcombo.DataSource = EvalTypes;
                }
                else
                {
                    dcombo.Items.Clear();
                    dcombo.Items.Add("Select a Measurement Type...");
                }
                dcombo.SelectedIndex = 0;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private DataTable getMeasureType()
        {
            DataTable retorno = null;
            try
            {
                string sql = "select ID_MEASURE, MEASURE_DESCRIPTION FROM DQ_MEASURES";
                retorno = _dbUtil.getItemsFromDatabase(sql);
                return retorno;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally
            {
                retorno = null;
            }
        }

        private int getComboSelectedValue(ComboBox combo)
        {
            int retorno = -1;
            try
            {
                if (combo.SelectedIndex > 0)
                {
                    bool result = int.TryParse(combo.SelectedValue.ToString(), out retorno);
                }
                return retorno;
            }
            catch (Exception ex)
            {
```

```
                MessageBox.Show(ex.Message.ToString(), "Error");
                return -1;
            }
        }

        private void setComboBoxValue(ComboBox combo, int Value)
        {
            try
            {
                if (Value > -1)
                {
                    combo.SelectedValue = (object)Value;
                }
                else
                {
                    combo.SelectedIndex = 0;
                }

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void setComboBoxIndex(ComboBox combo, int Index)
        {
            try
            {
                combo.SelectedIndex = Index;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadExistingInfo()
        {
            ExpectedCompletenessRecord ecr = null;
            try {
                if (this._qr != null)
                {
                    if (this._qr.expectedQuality != null)
                    {
                        ecr = (ExpectedCompletenessRecord)this._qr.expectedQuality;
                        int measureType = ecr.MeasureType;
                        int accept = ecr.acceptOption.GetHashCode();
                        int complAssessment = ecr.completenessAssessment.GetHashCode();
                        int cOperator = ecr.COperator.GetHashCode();
                        string v = ecr.Value;
                        string vu = ecr.ValueUnit;

                        setComboBoxValue(this.cboDQ_MEASUREDESC, measureType);
                        setComboBoxIndex(this.cboOperator, cOperator);
                        setComboBoxIndex(this.cboPassFail, accept);
                        this.txtDQ_Value.Text = v;
                        this.txtDQ_ValueUnit.Text = vu;
                    }

                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                ecr = null;
            }
        }

        private ExpectedCompletenessRecord buildReturn(AcceptReject ar, CompletenessAssessment ca, int
mt, CompareOperator co, string value, string valueUnit)
        {
            ExpectedCompletenessRecord retorno = null;
```

```
            try {
                retorno = new ExpectedCompletenessRecord();
                retorno.acceptOption = ar;
                retorno.completenessAssessment = ca;
                retorno.MeasureType = mt;
                retorno.COperator = co;
                retorno.Value = value;
                retorno.ValueUnit = valueUnit;

                return retorno;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
                return null;
            }
            finally {
                retorno = null;
            }
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            ExpectedCompletenessRecord ecr = null;
            try {
                AcceptReject ar = AcceptReject.undefined;
                CompletenessAssessment ca = CompletenessAssessment.undefined;
                CompareOperator co = CompareOperator.undefined;
                int mt = -1;
                string value = string.Empty;
                string valueunit = string.Empty;

                ar = (AcceptReject)this.cboPassFail.SelectedIndex;
                ca = CompletenessAssessment.CompletenessRecord;
                co = (CompareOperator)this.cboOperator.SelectedIndex;
                mt = getComboSelectedValue(this.cboDQ_MEASUREDESC);
                value = this.txtDQ_Value.Text;
                valueunit = this.txtDQ_ValueUnit.Text;

                ecr = buildReturn(ar, ca, mt, co, value, valueunit);

                if (ecr != null)
                {
                    this._qr.expectedQuality = (ExpectedQ)ecr;
                    this.DialogResult = System.Windows.Forms.DialogResult.OK;
                    this.Hide();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                ecr = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO;
using ISOSpatialQuality.Utilities;

namespace ISOSpatialQuality.Forms
{
    public partial class frmUserExpectedFormat : Form
```

```
    {
        private ExpectedFormatConsistencyCAD _expectedFormat = null;

        public ExpectedFormatConsistencyCAD expectedFormat { get { return _expectedFormat; } set { _exp
ectedFormat = value; } }

        private void loadExpectedFormat()
        {
            try {
                if (expectedFormat == null)
                {
                    this.cboFileFormat.SelectedIndex = 0;
                    this.cboPassFail.SelectedIndex = 0;
                }
                else
                {
                    this.cboFileFormat.SelectedIndex = this._expectedFormat.fileFormat.GetHashCode();
                    this.cboPassFail.SelectedIndex = this._expectedFormat.acceptOption.GetHashCode();
                }
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        public frmUserExpectedFormat()
        {
            InitializeComponent();
        }

        private void cmdOk_Click(object sender, EventArgs e)
        {
            try {
                if (_expectedFormat == null)
                {
                    _expectedFormat = new ExpectedFormatConsistencyCAD();
                }
                _expectedFormat.fileFormat = (FileFormat)this.cboFileFormat.SelectedIndex;
                _expectedFormat.acceptOption = (AcceptReject)this.cboPassFail.SelectedIndex;
                this.DialogResult = System.Windows.Forms.DialogResult.OK;
                this.Hide();
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadFileFormatCombo(ComboBox combo)
        {
            try {
                combo.DataSource = Enum.GetValues(typeof(FileFormat));
                combo.SelectedIndex = 0;
            }
            catch (Exception ex) {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void loadAcceptReject(ComboBox combo)
        {
            try {
                combo.DataSource = Enum.GetValues(typeof(AcceptReject));
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Error");
            }
        }

        private void frmUserExpectedFormat_Load(object sender, EventArgs e)
        {
            loadFileFormatCombo(this.cboFileFormat);
            loadAcceptReject(this.cboPassFail);
            loadExpectedFormat();
```

```
        }

        private void cmdCancel_Click(object sender, EventArgs e)
        {
            this.DialogResult = System.Windows.Forms.DialogResult.Cancel;
            this.Hide();
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO._19103;

namespace ISOSpatialQuality.Classes.ISO._19109
{
    public class GF_AttributeType : GF_PropertyType
    {
        private TypeName _valueType = null;
        private string _domainOfValues = string.Empty;

        public TypeName valueType
        {
            get { return _valueType; }
            set { _valueType = value; }
        }

        public string domainOfValues
        {
            get { return _domainOfValues; }
            set { _domainOfValues = value; }
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO._19103;

namespace ISOSpatialQuality.Classes.ISO._19109
{
    public class GF_FeatureType
    {
        private LocalName _typeName = null;
        private bool _isAbstract = false;
        private string _definition = string.Empty;

        public LocalName typeName
        {
            get { return _typeName; }
            set { _typeName = value; }
        }

        public string definition
        {
            get { return _definition; }
            set { _definition = value; }
        }

        public bool isAbstract
        {
            get { return _isAbstract; }
            set { _isAbstract = value; }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO._19103;

namespace ISOSpatialQuality.Classes.ISO._19109
{
    public class GF_Operation : GF_PropertyType
    {
        private string _signature = string.Empty;

        public string signature {
            get {
                return _signature;
            }
            set {
                _signature = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using ISOSpatialQuality.Classes.Codelists;
using ISOSpatialQuality.Classes.ISO._19103;

namespace ISOSpatialQuality.Classes.ISO._19109
{
    public abstract class GF_PropertyType
    {
        private LocalName _memberName = null;
        private string _definition = null;

        public LocalName memberName {
            get {
                return _memberName;
            }
            set {
                _memberName = value;
            }
        }
        public string definition {
            get {
                return _definition;
            }
            set {
                _definition = value;
            }
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class MD_ScopeDescription
    {
        private string _feature = string.Empty;
        private string _attributes = string.Empty;
        private string _dataset = string.Empty;
        private string _other = string.Empty;
```

```
        public string feature
        {
            get { return _feature; }
            set { _feature = value; }
        }

        public string attributes
        {
            get { return _attributes; }
            set { _attributes = value; }
        }

        public string dataset
        {
            get { return _dataset; }
            set { _dataset = value; }
        }

        public string other
        {
            get { return _other; }
            set { _other = value; }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality;
using ISOSpatialQuality.Classes;
using ISOSpatialQuality.Forms;

namespace ISOSpatialQuality.MDL
{
    internal class ISOMain
    {
        public static void ExecuteMainForm(System.String unparsed)
        {
            frmQualitySetup mainForm = null;
            try {
                mainForm = new frmQualitySetup();
                mainForm.Show();
            }
            catch (Exception ex) {
                System.Windows.Forms.MessageBox.Show(ex.Message.ToString(), "Error");
            }
            finally {
                mainForm = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Data;
using System.Xml.Linq;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class LI_Lineage
    {
        private string _projectionSystem = string.Empty;
        private string _datum = string.Empty;
        private string _ellipsoid = string.Empty;
        private string _scale = string.Empty;
        private HistoryCollection _hc = null;

        public LI_Lineage()
```

```
        {
            _hc = new HistoryCollection();
        }
        public string ProjectionSystem { get { return _projectionSystem; } set { _projectionSystem = va
lue; } }
        public string Datum { get { return _datum; } set { _datum = value; } }
        public string Ellipsoid { get { return _ellipsoid; } set { _ellipsoid = value; } }
        public string Scale { get { return _scale; } set { _scale = value; } }
        public HistoryCollection History { get { return _hc; } set { _hc = value; } }

        public XElement GetXMLInfo()
        {
            XElement retorno = null;
            try {
                XElement processStep = History.GetXMLHistory();
                retorno = new XElement("source",
                    new XAttribute("scaleDenominator", this.Scale),
                    new XAttribute("sourceReferenceSystem", this.ProjectionSystem),
                    new XAttribute("sourceDatum", this.Datum),
                    new XAttribute("sourceEllipsoid", this.Ellipsoid));
                retorno.Add(processStep);
                return retorno;
            }
            catch (Exception ex) {
                return null;
                throw new Exception("Error", ex);
            }
            finally {
                retorno = null;
            }
        }

        private string getAttributeValue_string(XAttribute stepInfo)
        {
            string retorno = string.Empty;
            try {
                if (stepInfo != null)
                {
                    retorno = stepInfo.Value.ToString();
                }
                return retorno;
            }
            catch (Exception ex) {
                return string.Empty;
                throw new Exception("Error", ex);
            }
        }

        private LI_History getHistoryFromXML(XElement processStep)
        {
            LI_History retorno = null;
            try {
                if (processStep != null)
                {
                    XAttribute xdata = processStep.Attribute("dateTime");
                    string data = getAttributeValue_string(xdata);
                    XAttribute xdesc = processStep.Attribute("description");
                    string desc = getAttributeValue_string(xdesc);

                    DateTime dt = DateTime.Today;
                    bool result = DateTime.TryParse(data, out dt);
                    if (result)
                    {
                        retorno = new LI_History();
                        retorno.DateSource = dt;
                        retorno.History = desc;
                    }
                }
                return retorno;
            }
            catch (Exception ex) {
                return null;
                throw new Exception("Error", ex);
            }
```

```
            finally { retorno = null; }
        }

        public LI_Lineage(string XMLFile)
        {
            XDocument xdoc = null;
            try {
                _hc = new HistoryCollection();
                if (!string.IsNullOrEmpty(XMLFile))
                {
                    xdoc = XDocument.Load(XMLFile);
                    if (xdoc != null)
                    {
                        XElement source = xdoc.Element("DQ_DataQuality").Element("lineage").Element("so
urce");
                        if (source != null)
                        {
                            XElement pSteps = source.Element("processSteps");

                            if (pSteps != null)
                            {
                                IEnumerable<XElement> processSteps = from el in pSteps.Elements("Proces
sStep") select el;
                                foreach (XElement processStep in processSteps)
                                {
                                    LI_History lih = getHistoryFromXML(processStep);
                                    if (lih != null)
                                    {
                                        _hc.addHistory(lih);
                                    }
                                }
                            }


                            XAttribute xps = source.Attribute("sourceReferenceSystem");
                            string ps = getAttributeValue_string(xps);

                            XAttribute xdt = source.Attribute("sourceDatum");
                            string dt = getAttributeValue_string(xdt);

                            XAttribute xel = source.Attribute("sourceEllipsoid");
                            string ell = getAttributeValue_string(xel);

                            XAttribute xsc = source.Attribute("scaleDenominator");
                            string sc = getAttributeValue_string(xsc);

                            this._scale = sc;
                            this._ellipsoid = ell;
                            this._datum = dt;
                            this._projectionSystem = ps;
                        }


                    }
                }
            }
            catch (Exception ex) {
                throw new Exception("Error", ex);
            }
            finally { }
        }
    }

    public class LI_History
    {
        private string _history = string.Empty;
        private DateTime _dateSource = DateTime.Today;

        public string History {get {return _history;} set {_history = value;}}
        public DateTime DateSource {get {return _dateSource;}set {_dateSource = value;}}
    }

    public class HistoryCollection : CollectionBase
    {
```

```
        public int addHistory(LI_History property)
        {
            return (List.Add(property));
        }
        public void InsertAt(LI_History property, int Index)
        {
            List.Insert(Index, property);
            List.RemoveAt(Index);
        }
        public void removeHistory(LI_History property)
        {
            List.Remove(property);
        }
        public int IndexOf(LI_History property)
        {
            return (List.IndexOf(property));
        }
        public LI_History this[int Index]
        {
            get
            {
                return (LI_History)List[Index];
            }
            set
            {
                List[Index] = value;
            }
        }

        private DataColumn buildColumn(string fieldType, string FieldName, bool allowDBNull, int MaxLen
gth, bool isUnique, bool IsAutoIncremented, long AutoIncrementSeed, long AutoIncrementedSeep)
        {
            DataColumn retorno = null;
            try
            {
                retorno = new DataColumn();
                retorno.DataType = System.Type.GetType(fieldType);
                retorno.ColumnName = FieldName;
                retorno.Caption = FieldName;
                retorno.AllowDBNull = allowDBNull;
                retorno.MaxLength = MaxLength;
                retorno.Unique = isUnique;
                retorno.AutoIncrement = IsAutoIncremented;
                retorno.AutoIncrementSeed = AutoIncrementSeed;
                retorno.AutoIncrementStep = AutoIncrementedSeep;
                return retorno;
            }
            catch (Exception ex)
            {
                return null;
                throw new Exception("Error", ex);
            }
            finally
            {
                retorno.Dispose();
                retorno = null;
            }
        }

        private DataTable BuildDatatable()
        {
            DataTable retorno = null;

            try {
                DataColumn dcData = buildColumn("System.DateTime", "HISTORY_DATE", true, -
1, false, false, -1, -1);
                DataColumn dcHist = buildColumn("System.String", "HISTORY", true, 1024, false, false, -
1, -1);
                retorno = new DataTable();
                retorno.Columns.Add(dcData);
                retorno.Columns.Add(dcHist);
                return retorno;
            }
            catch (Exception ex) {
```

```
                return null;
                throw new Exception("Error", ex);
            }
            finally {
                retorno = null;
            }
        }

        public DataTable GetDatatable()
        {
            DataTable retorno = null;
            try {
                retorno = BuildDatatable();

                for (int i = 0; i < this.Count; i++)
                {
                    LI_History lih = this[i];
                    DataRow drow = retorno.NewRow();
                    drow["HISTORY_DATE"] = lih.DateSource;
                    drow["HISTORY"] = lih.History;
                    lih = null;
                    retorno.Rows.Add(drow);
                }

                return retorno;
            }
            catch (Exception ex) {
                return null;
                throw new Exception("Error", ex);
            }
            finally {
                retorno = null;
            }
        }

        public XElement GetXMLHistory()
        {
            XElement retorno = null;
            try {

                retorno = new XElement("processSteps");

                for (int i = 0; i < this.Count; i++)
                {
                    LI_History lih = this[i];
                    string data = lih.DateSource.ToShortDateString();
                    string description = lih.History;
                    XElement processStep = new XElement("ProcessStep",
                        new XAttribute("dateTime", data),
                        new XAttribute("description", description));

                    retorno.Add(processStep);
                }

                return retorno;
            }
            catch (Exception ex) {
                return null;
                throw new Exception("Error", ex);
            }
            finally {
                retorno = null;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19103
{
```

```
    public class LocalName : GenericName
    {
        public virtual string aName { get; set; }
    }
}

=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ISOSpatialQuality.Classes.ISO._19115
{
    public class MD_Identifier
    {
        private CI_Citation _authority = null;
        private string _code = string.Empty;

        public virtual CI_Citation authority
        {
            get
            {
                return _authority;
            }
            set
            {
                _authority = value;
            }
        }

        public virtual string code
        {
            get
            {
                return _code;
            }
            set
            {
                _code = value;
            }
        }
    }
}
=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ISOSpatialQuality.Utilities;
using System.Data;

namespace ISOSpatialQuality.Classes.Codelists
{
    public class MD_ScopeCode
    {
        private Dictionary<string, string> _md_ScopeCode = null;
        Dictionaries buildDictionary = null;
        DBUtils dbUtils = null;

        public Dictionary<string, string> ScopeCode {
            get {
                DataTable scopeCode = dbUtils.getItemsFromDatabase("select MD_Scopecode, ScopeCd from M
D_SCOPECODE");
                _md_ScopeCode = buildDictionary.buildDictionary(scopeCode, "ScopeCd", "MD_Scopecode");
                return _md_ScopeCode;
            }
            set {
                _md_ScopeCode = value;
            }
        }

        public MD_ScopeCode()
        {
```

```
            _md_ScopeCode = new Dictionary<string, string>();
            buildDictionary = new Dictionaries();
            dbUtils = new DBUtils();
        }

        ~MD_ScopeCode()
        {
            _md_ScopeCode = null;
            dbUtils = null;
            buildDictionary = null;
        }
    }
}
=====================================================================
<?xml version="1.0" encoding="utf-8" ?>
<KeyinTree xmlns="http://www.bentley.com/schemas/1.0/MicroStation/AddIn/KeyinTree.xsd">

    <RootKeyinTable ID="root">
        <Keyword SubtableRef="ISOSpatialQuality" CommandClass="MacroCommand" CommandWord="ISOSpatialQua
lity" >
            <Options Required="true"/>
        </Keyword>
    </RootKeyinTable>

    <SubKeyinTables>
        <KeyinTable ID="ISOSpatialQuality">
            <Keyword CommandWord="Start"> </Keyword>
        </KeyinTable>
    </SubKeyinTables>

    <KeyinHandlers>
        <KeyinHandler Keyin="ISOSpatialQuality Start" Function="ISOSpatialQuality.MDL.MyKeyinCommands.I
nitApp"/>
    </KeyinHandlers>

</KeyinTree>

=====================================================================
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace ISOSpatialQuality.MDL
{
    internal class MyKeyinCommands
    {
        public static void InitApp(System.String unparsed)
        {
            Application.EnableVisualStyles();
            ISOMain.ExecuteMainForm(unparsed);
        }
    }
}
=====================================================================
```