OSIsoft.

vCampus Live! 2011

PI Event Frames for Corrosion Coupon Data

By: Randy Esposito & Rick Davin

Nalco Energy Services – Automation Center of Excellence

Agenda

- Who is Nalco, what do we do
- What is a “Corrosion Coupon” and why do I care?
- How can I model Corrosion Coupon data?
- How can I visualize this data?
- How can we transform the data into something usable?

Nalco Company:
Essential Expertise for Water, Energy, and Air℠

- World's leading process improvement company
- 70,000 customers in more than 130 countries
- 75 years of experience in the hydrocarbon industries

What is a Corrosion Coupon?

- Representative Metal sample sitting in the process stream for a given time (days to months)
- “Experiences” all of the process variations over it’s time in the system

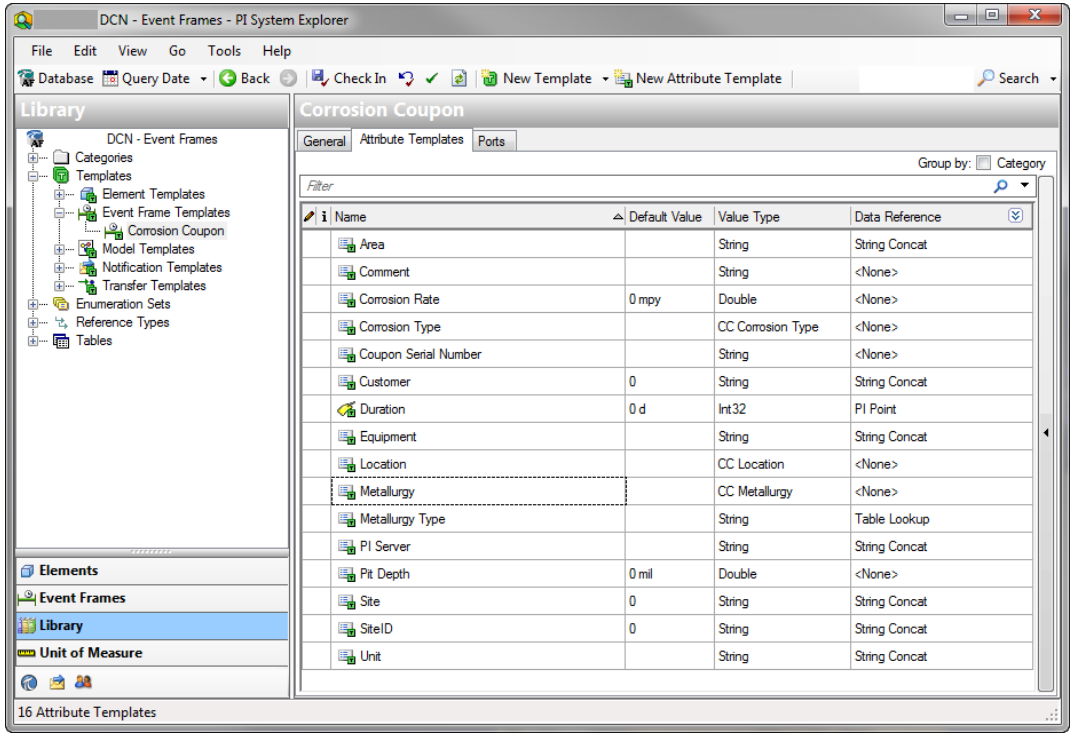
What data do we collect?

- “Static” Information about the coupon
  - Serial Number, Size, Metal Type, Physical Location
- “Process” Information
  - Date Inserted, Date Removed, Initial Weight, Final Weight, Calculated Corrosion Rate, Corrosion Type, Pit Depth

How Can we model this data?

- PI Tags
- Transactional data system (i.e SQL)
- PI Batch
- PI AF Tables
- **PI Event Frames**

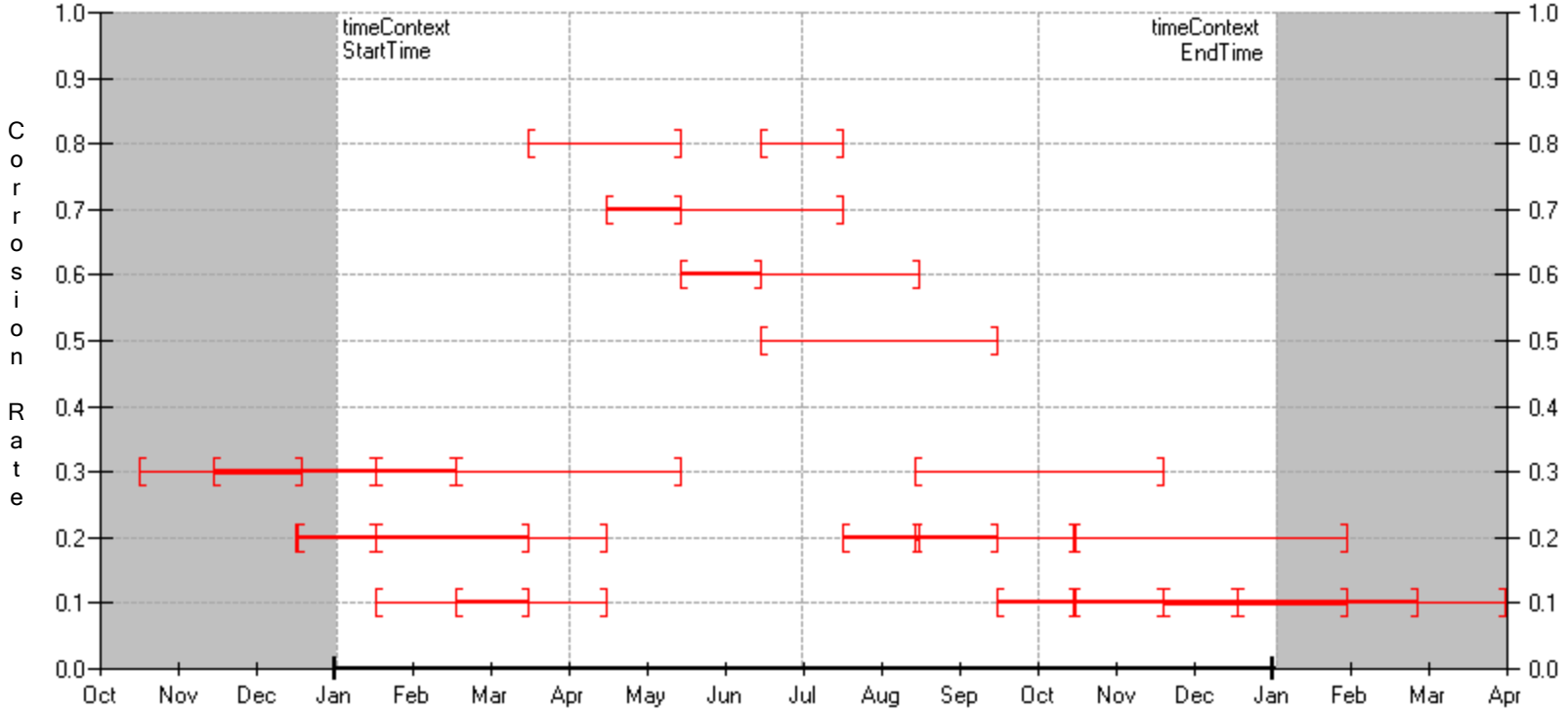
Corrosion Coupon PI Event Frame Template



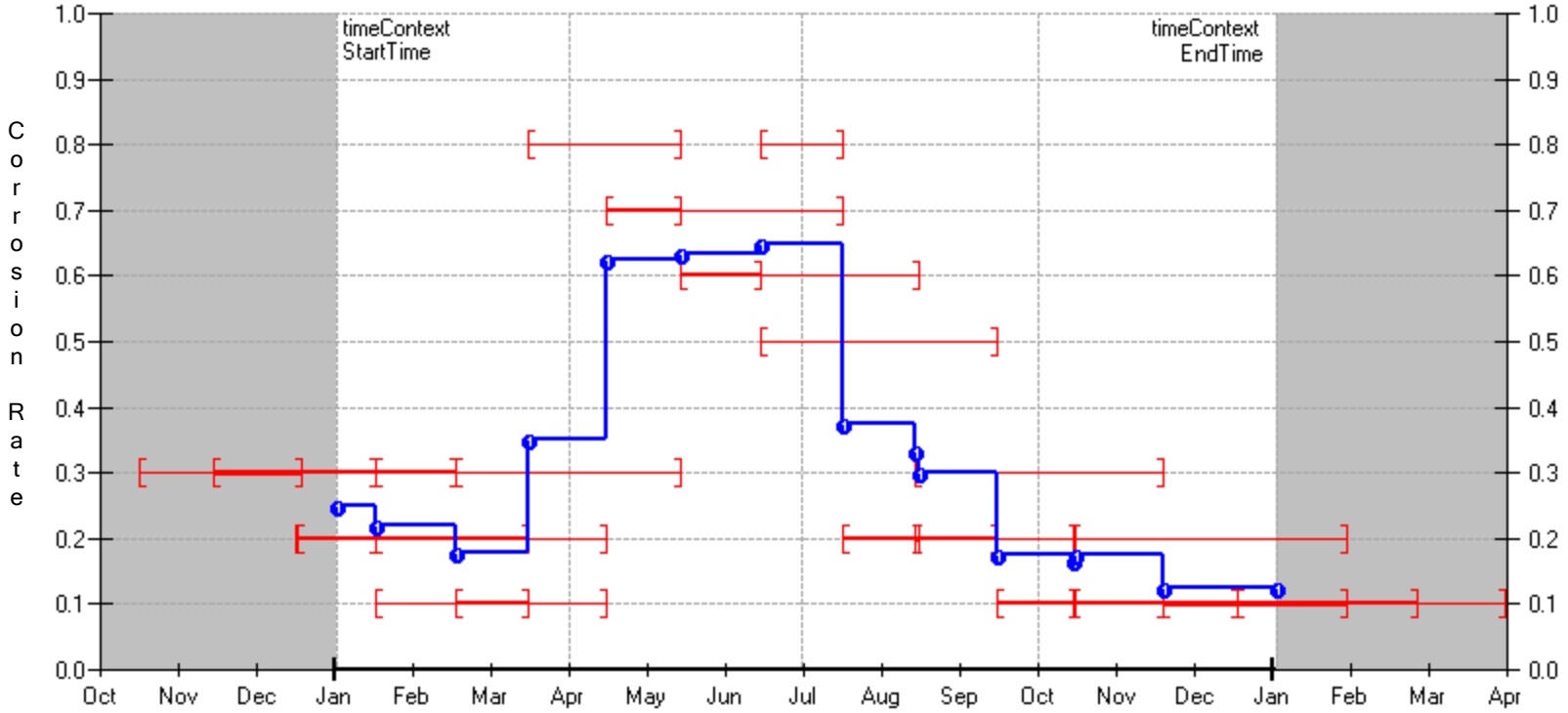
Key Lessons

- Event frame names don't have to be unique
- Time range endpoints need explicit handling
- Today you have to create your own data entry method (will change in future)

How to Visualize the Data?



How to Visualize the Data?



Nalco Corrosion Rate Data Reference

- Fetch all Corrosion Coupon event frames for a given time context.
- Return an average of the Corrosion Rate attribute at applicable event times.
- Build a list of applicable event times!

Fetching Event Frames

The fetching of the event frames data is the responsibility of the developer, which is a good thing:

- Better control of paged result sets
- Which specific search method is employed to match up with your application requirements
- Whether or not to include the StartTime and/or EndTime in your calculation.

AFSearchMode Enumeration

"Note that object values which end on the search start time or start on the search end time are not included as part of the returned collection."

- Applies when searching on a time window.
- Doesn't apply when searching on a start time and stepping off *N* count in some direction.

Your Application Behavior

Depends upon your business requirements.

Will your data …

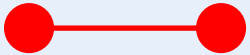
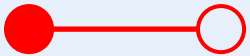
- ever be overlapped?
- ever have gaps in it?
- include or exclude the StartTime?
- include or exclude the EndTime?

Nalco's Corrosion Coupon Application

- Can never assume there will be exactly 1 event frame returned for an event timestamp.
- There can be 0, 1, or more than 1 event frames returned for any given event timestamp.
- Exclude StartTimes but include EndTimes.

Treatment of End Points

Nalco Corrosion Coupons

Icon	StartTime	EndTime
	Excluded	Excluded
	Included	Included
	Excluded	Included
	Included	Excluded

Suggestion: if searching on a fixed time window, you may consider extending the time window by 1-second on each end point.

The GetInputs Method

C# Example:

```
public override AFAttributeList GetInputs(object context)
{
     return null;
}
```

Visual Basic Example:

```
Public Overloads Overrides Function GetInputs(ByVal context As Object) As AFAttributeList
        Return Nothing
End Function
```

GetValues Method Signature

C# Example:

```
AFValues GetValues(object context,
                   AFTimeRange timeContext,
                   int numberOfValues,
                   AFAttributeList inputAttributes,
                   AFValues[] inputValues)
```

Visual Basic Example:

```
Function GetValues(ByVal context As Object,
                   ByVal timeContext As AFTimeRange,
                   ByVal numberOfValues As Integer,
                   ByVal inputAttributes As AFAttributeList,
                   ByVal inputValues As AFValues()
                   ) As AFValues
```

Turn Nothing Into Something

- Both **inputAttributes** and **inputValues** return nothing since **GetInputs** returned nothing.
- You fetch the event frames.
- You build the list of output events.
- You calculate a value (e.g. an average) for each output event.

BuildEventTimes (C#)

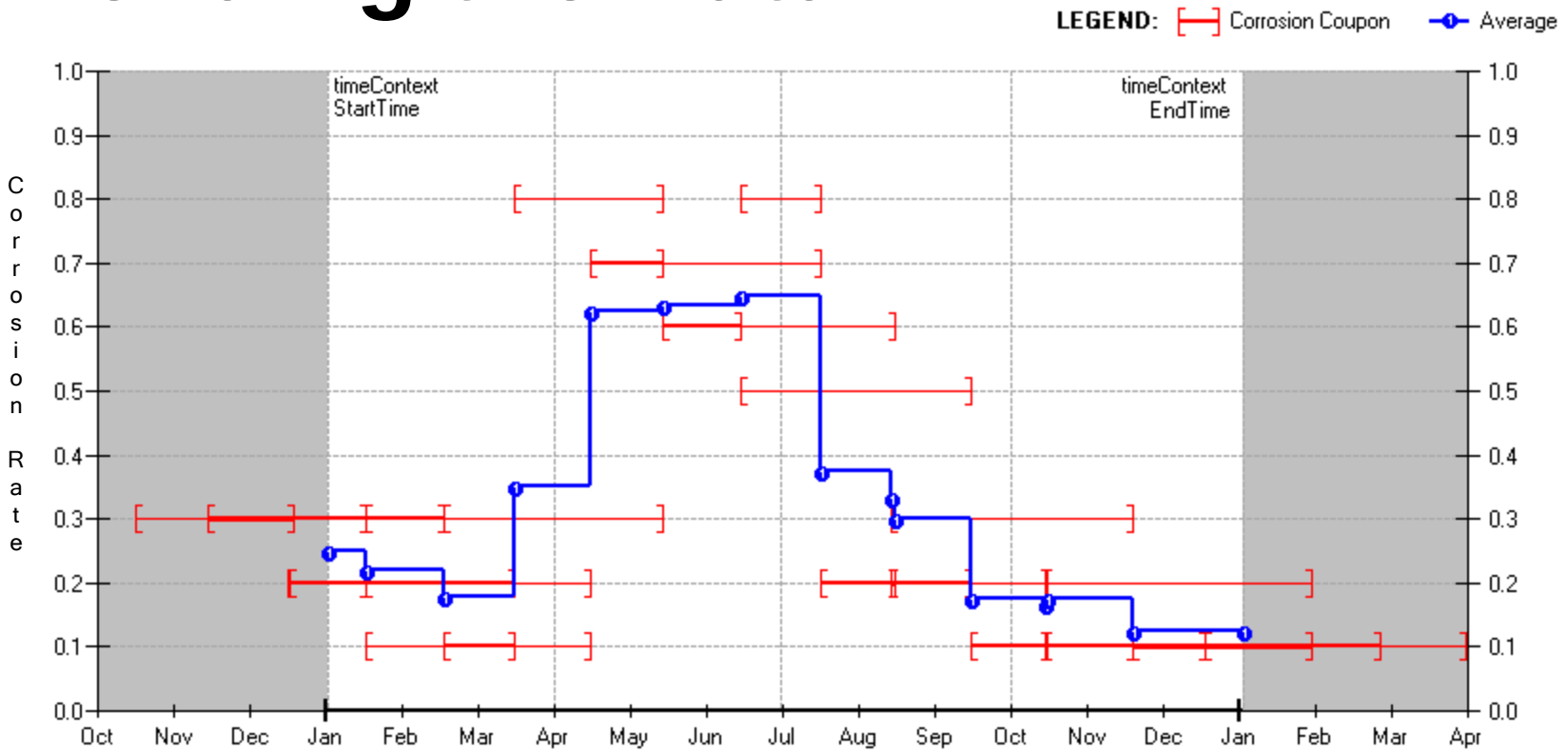
```
private List<AFTime> BuildEventTimes(AFTimeRange timeContext, AFNamedCollection<AFEventFrame> Frames)

{
    List<AFTime> Result = new List<AFTime>();
    AFTime t = default(AFTime);
    // First and foremost, we add the timeContext's StartTime
    Result.Add(timeContext.StartTime);
    if (Frames != null && Frames.Count > 0)
    {
        foreach (AFEventFrame frame in Frames)
        {
            t = new AFTime(frame.StartTime.UtcSeconds + 1);
            if (t > timeContext.StartTime)
            {
                if (!Result.Contains(t))
                    Result.Add(t);
            }
            t = new AFTime(frame.EndTime.UtcSeconds + 1);
            if (t < timeContext.EndTime)
            {
                if (!Result.Contains(t))
                    Result.Add(t);
            }
        }
        Result.Sort();
    }
    // Lastly, we add the timeContext's EndTime if it's not already there.
    if (!Result.Contains(timeContext.EndTime))
        Result.Add(timeContext.EndTime);
    return Result;
}
```

BuildEventTimes (VB)

```
Private Function BuildEventTimes(ByVal timeContext As AFTimeRange,
                                 ByVal Frames As AFNamedCollection(Of AFEventFrame)
                                 ) As List(Of AFTime)
    Dim Result As New List(Of AFTime)
    Dim t As AFTime
    ' First and foremost, we add the timeContext's StartTime
    Result.Add(timeContext.StartTime)
    If Frames IsNot Nothing AndAlso Frames.Count > 0 Then
        For Each frame As AFEventFrame In Frames
            t = New AFTime(frame.StartTime.UtcSeconds + 1)
            If t > timeContext.StartTime Then
                If Not Result.Contains(t) Then Result.Add(t)
            End If
            t = New AFTime(frame.EndTime.UtcSeconds + 1)
            If t < timeContext.EndTime Then
                If Not Result.Contains(t) Then Result.Add(t)
            End If
        Next
        Result.Sort()
    End If
    ' Lastly, we add the timeContext's EndTime if it's not already there.
    If Not Result.Contains(timeContext.EndTime) Then Result.Add(timeContext.EndTime)
    Return Result
End Function
```

Trending the Data



Conclusions

We talked about:

- How PI Event Frames make it easy to represent Corrosion Coupon Data
- How to visualize the event frame data
- How to create a PI AF data reference to transform the event frame information into a time series function.

What else can we apply these methods to????

Thank you